사용자 가이드

AWS Amplify 호스팅

AWS Amplify 호스팅: 사용자 가이드

Table of Contents

AWS Amplify 호스팅 시작

Amplify Hosting은 지속적인 배포로 풀 스택 서버리스 웹 애플리케이션을 호스팅하기 위한 Git 기반 워크플로를 제공합니다. Amplify는 AWS 글로벌 콘텐츠 전송 네트워크(CDN)에 앱을 배포합니다. 이 사용 설명서는 Amplify Hosting을 시작하는 데 필요한 정보를 제공합니다.

지원되는 프레임워크

Amplify Hosting은 다음을 포함하여 일반적인 SSR 프레임워크, 단일 페이지 애플리케이션(SPA) 프레임워크 및 정적 사이트 생성기를 다수 지원합니다.

SSR 프레임워크

- Next.js
- Nuxt
- Astro 커뮤니티 어댑터 사용
- SvelteKit 커뮤니티 어댑터 사용
- 사용자 지정 어댑터를 사용하는 모든 SSR 프레임워크

SPA 프레임워크

- React
- Angular
- Vue.js
- Ionic
- Ember

정적 사이트 생성기

- Eleventy
- Gatsby
- Hugo
- Jekyll
- VuePress

Amplify Hosting 기능

기능 브랜치

새로운 브랜치를 연결하여 프런트엔드 및 백엔드용 프로덕션 및 스테이징 환경을 관리합니다.

사용자 지정 도메인

애플리케이션을 사용자 지정 도메인에 연결합니다.

풀 요청 미리 보기

코드 검토 중에 변경 사항을 미리 봅니다.

엔드 투 엔드 테스트

엔드 투 엔드 테스트로 앱 품질을 개선합니다.

암호로 보호되는 브랜치

암호로 웹 애플리케이션을 보호하므로 공개적으로 액세스할 수 없어도 새로운 기능을 사용할 수 있습니다.

리디렉션 및 다시 쓰기

다시 쓰기 및 리디렉션을 설정하여 SEO 순위를 유지하고 클라이언트 앱 요구 사항에 따라 트래픽을 라우팅합니다.

원자 배포

원자 배포는 전체 배포가 완료된 후에만 웹 애플리케이션이 배포되도록 함으로써 유지 관리 기간을 없앱니다. 이렇게 하면 파일을 제대로 업로드하지 못하는 시나리오가 제거됩니다.

Amplify Hosting 시작하기

Amplify Hosting을 시작하려면 Amplify Hosting에 앱 배포 시작하기 자습서를 참조하세요. 자습서를 완료한 후에는 Git 리포지토리(GitHub, BitBucket, GitLab 또는 AWS CodeCommit)에서 웹 앱을 연결하고 지속적인 배포를 통해 Amplify Hosting에 배포하는 방법을 알게 됩니다.

백엔드 빌드

AWS Amplify Gen 2는 백엔드를 정의하기 위한 TypeScript 기반 코드 우선 개발자 경험을 도입했습니다. Amplify Gen 2를 사용하여 백엔드를 빌드하고 앱에 연결하는 방법을 알아보려면 Amplify 설명서의 백엔드 빌드 및 연결을 참조하세요.

Amplify Gen 2의 코드 우선 접근 방식을 더 잘 이해하려면 AWS Workshop Studio 웹 사이트의 Amplify Gen 2 워크숍을 참조하세요. 이 포괄적인 자습서에서는 React 및 Next.js를 사용하여 서버리스 애플리케이션을 빌드하고 Amplify Gen 2 Data 및 Auth 라이브러리와 Amplify UI 라이브러리를 사용하여 애플리케이션에 기능을 추가하는 방법을 알아봅니다.

CLI 및 Amplify Studio를 사용하여 Gen 1 앱의 백엔드를 빌드하기 위한 설명서를 찾고 있는 경우 Gen 1 Amplify 설명서의 백엔드 빌드 및 연결을 참조하세요.

Amplify Hosting 요금

AWS Amplify 는 사용한 만큼만 요금을 청구합니다. 자세한 설명은 AWS Amplify 요금을 참조하세요.

Amplify Hosting에 앱 배포 시작하기

다음 자습서에서는 Amplify Hosting의 작동 방식을 이해하는 데 도움이 되도록 Amplify가 지원하는 일반적인 SSR 프레임워크를 사용하여 생성된 애플리케이션을 빌드하고 배포하는 방법을 안내합니다.

자습서

- Amplify Hosting에 Next.js 앱 배포
- Amplify Hosting에 Nuxt.js 앱 배포
- Amplify Hosting에 Astro.js 앱 배포
- Amplify Hosting에 SvelteKit 앱 배포

Amplify Hosting에 Next.js 앱 배포

이 자습서에서는 Git 리포지토리에서 Next.js 애플리케이션을 빌드하고 배포하는 방법을 안내합니다.

이 자습서를 시작하기 전에 다음 사전 조건을 완료합니다.

에 가입 AWS 계정

아직 AWS 고객이 아닌 경우 온라인 지침에 따라 를 생성 AWS 계정해야 합니다. 가입하면 Amplify 및 애플리케이션에 사용할 수 있는 기타 AWS 서비스에 액세스할 수 있습니다.

애플리케이션 만들기

Next.js 설명서의 create-next-app 명령을 사용하여 이 자습서에 사용할 기본 Next.js 애플리케이션을 생성합니다.

Git 리포지토리 생성

Amplify는 GitHub, Bitbucket, GitLab 및를 지원합니다 AWS CodeCommit. `create-next-app` 애플리케이션을 Git 리포지토리로 푸시합니다.

1단계: Git 리포지토리 연결

이 단계에서는 Git 리포지토리의 Next.js 애플리케이션을 Amplify Hosting에 연결합니다.

Git 리포지토리의 앱을 연결하려면

1. Amplify 콘솔을 엽니다.

2. 현재 리전에서 처음 앱을 배포하는 경우 기본적으로 AWS Amplify 서비스 페이지에서 시작합니다.

   페이지 상단에서 앱 배포를 선택합니다.

3. Amplify로 빌드 시작 페이지에서 Git 리포지토리 공급자를 선택하고 다음을 선택합니다.

   GitHub 리포지토리의 경우, Amplify는 GitHub 앱 기능을 사용하여 Amplify에 액세스 권한을 부여합니다. GitHub 앱 설치 및 인증에 대한 자세한 내용은 GitHub 리포지토리에 대한 Amplify 액세스 설정을 참조하십시오.

   > **Note**
   >
   > Bitbucket, GitLab 또는를 사용하여 Amplify 콘솔에 권한을 부여하면 AWS CodeCommit Amplify는 리포지토리 공급자로부터 액세스 토큰을 가져오지만 토큰은 서버에 저장되지 않습니다. AWS Amplify는 특정 리포지토리에만 설치된 배포 키를 사용하여 리포지토리에 액세스합니다.

4. 리포지토리 브랜치 추가 페이지에서 다음을 수행하십시오.

   a. 연결할 리포지토리의 이름을 선택합니다.

   b. 연결할 리포지토리 브랜치의 이름을 선택합니다.

   c. 다음을 선택합니다.

2단계: 빌드 설정 확인

Amplify는 배포 중인 브랜치에 대해 실행할 빌드 명령 시퀀스를 자동으로 감지합니다. 이 단계에서는 빌드 설정을 검토하고 확인합니다.

앱의 빌드 설정을 확인하려면

1. 앱 설정 페이지에서 빌드 설정 섹션을 찾습니다.

   프론트엔드 빌드 명령 및 빌드 출력 디렉터리가 올바른지 확인합니다. 이 Next.js 예제 앱의 경우 빌드 출력 디렉터리가 `.next`로 설정되어 있습니다.

2. 서비스 역할을 추가하는 절차는 새 역할을 생성할지 아니면 기존 역할을 사용할지에 따라 달라집니다.

   - 새 역할을 생성하려면

     - 새 서비스 역할 생성 및 사용을 선택합니다.

- 기존 역할을 사용하려면

  a. 기존 역할 사용을 선택합니다.

  b. 서비스 역할 목록에서 사용할 역할을 선택합니다.

3. 다음을 선택합니다.

3단계: 애플리케이션 배포

이 단계에서는 AWS 글로벌 콘텐츠 전송 네트워크(CDN)에 앱을 배포합니다.

앱을 저장 및 배포하려면

1. 검토 페이지에서 리포지토리 세부 정보와 앱 설정이 올바른지 확인합니다.
2. 저장 및 배포를 선택합니다. 프론트엔드 빌드는 일반적으로 1~2 분이 소요되지만 앱 크기에 따라 다를 수 있습니다.
3. 배포가 완료되면 `amplifyapp.com` 기본 도메인에 대한 링크를 사용하여 앱을 볼 수 있습니다.

> **Note**
>
> Amplify 애플리케이션의 보안을 강화하기 위해 amplifyapp.com 도메인은 공개 서픽스 목록(PSL)에 등록되어 있습니다. 보안 강화를 위해 Amplify 애플리케이션의 기본 도메인 이름에 민감한 쿠키를 설정해야 하는 경우, `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도(CSRF)로부터 도메인을 보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

4단계: (선택 사항) 리소스 정리

자습서용으로 생성한 앱이 더 이상 필요 없는 경우에는 삭제할 수 있습니다. 이렇게 하면 사용하지 않는 리소스에 요금이 청구되지 않습니다.

앱을 삭제하려면

1. 탐색 창의 앱 설정 메뉴에서 일반 설정을 선택합니다.
2. 일반 설정 페이지에서 앱 삭제를 선택합니다.
3. 확인 창에서 **delete**를 입력합니다. 그런 다음 앱 삭제를 선택합니다.

앱에 기능 추가

Amplify에 앱을 배포했으므로 호스팅된 애플리케이션에서 사용할 수 있는 다음 기능 중 일부를 살펴볼 수 있습니다.

환경 변수

애플리케이션은 런타임에 구성 정보가 필요한 경우가 많습니다. 이러한 구성은 데이터베이스 연결 세부 정보, API 키 또는 파라미터가 될 수 있습니다. 환경 변수는 빌드 시 이러한 구성을 노출하는 수단을 제공합니다. 자세한 내용은 환경 변수를 참조하세요.

사용자 지정 도메인

이 자습서에서는 Amplify가 `https://branch-name.d1m7bkiki6tdw1.amplifyapp.com`과 같은 URL을 사용하여 기본 `amplifyapp.com` 도메인에서 앱을 호스팅합니다. 앱을 사용자 지정 도메인에 연결하면 사용자는 앱이 `https://www.example.com`과 같은 사용자 지정 URL에서 호스팅되는 것을 볼 수 있습니다. 자세한 내용은 사용자 지정 도메인 설정을 참조하세요.

풀 요청 미리 보기

풀 요청 웹 미리 보기는 팀이 코드를 프로덕션 또는 통합 브랜치에 병합하기 전에 pull 요청(PR)의 변경 사항을 미리 볼 수 있는 방법을 제공합니다. 자세한 내용은 풀 요청의 웹 미리 보기 섹션을 참조하세요.

여러 환경 관리

Amplify가 기능 브랜치 및 GitFlow 워크플로와 함께 작동하여 여러 배포를 지원하는 방법을 알아보려면 기능 브랜치 배포 및 팀 워크플로를 참조하세요.

Amplify Hosting에 Nuxt.js 앱 배포

Nuxt.js 애플리케이션을 Amplify Hosting에 배포하려면 다음 지침을 따릅니다. Nuxt는 Nitro 서버를 사용하여 사전 설정된 어댑터를 구현했습니다. 이를 통해 추가 구성 없이 Nuxt 프로젝트를 배포할 수 있습니다.

Amplify Hosting에 Nuxt 앱을 배포하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 리포지토리 공급자를 선택하고 다음을 선택합니다.

4. 리포지토리 브랜치 추가 페이지에서 다음을 수행합니다.

   a. 연결할 리포지토리의 이름을 선택합니다.

   b. 연결할 리포지토리 브랜치의 이름을 선택합니다.

   c. 다음을 선택합니다.

5. Amplify가 Amazon CloudWatch Logs에 앱 로그를 전달할 수 있도록 하려면 콘솔에서 명시적으로 활성화해야 합니다. 고급 설정 섹션을 열고 서버 측 렌더링(SSR) 배포 섹션에서 SSR 앱 로그 활성화를 선택합니다.
6. 다음을 선택합니다.
7. 검토 페이지에서 저장 및 배포를 선택합니다.

Amplify Hosting에 Astro.js 앱 배포

Astro.js 애플리케이션을 Amplify Hosting에 배포하려면 다음 지침을 따릅니다. 기존 애플리케이션을 사용하거나 Astro가 제공하는 공식 예제 중 하나를 사용하여 스타터 애플리케이션을 생성할 수 있습니다. 스타터 애플리케이션을 생성하려면 Astro 설명서의 테마 또는 스타터 템플릿 사용을 참조하세요.

SSR로 Astro 사이트를 Amplify Hosting에 배포하려면 애플리케이션에 어댑터를 추가해야 합니다. 당사는 Amplify 소유의 Astro 프레임워크용 어댑터를 유지 관리하지 않습니다. 이 자습서에서는 커뮤니티의 구성원이 생성한 `astro-aws-amplify` 어댑터를 사용합니다. 이 어댑터는 GitHub 웹 사이트의 github.com/alexnguyennz/astro-aws-amplify 사용할 수 있습니다.이 어댑터는 유지 관리하지 AWS 않습니다.

Amplify Hosting에 Astro 앱을 배포하려면

1. 로컬 컴퓨터에서 배포할 Astro 애플리케이션으로 이동합니다.
2. 어댑터를 설치하려면 터미널 창을 열고 다음 명령을 실행합니다. 이 예제에서는 github.com/alexnguyennz/astro-aws-amplify에서 사용할 수 있는 커뮤니티 어댑터를 사용합니다. *`astro-aws-amplify`*를 사용 중인 어댑터 이름으로 바꿀 수 있습니다.

   ```
   npm install astro-aws-amplify
   ```

3. Astro 앱의 프로젝트 폴더에서 `astro.config.mjs` 파일을 엽니다. 이 파일을 업데이트하여 어댑터를 추가합니다. 파일이 다음과 같아야 합니다.

   ```
   import { defineConfig } from 'astro/config';
   import mdx from '@astrojs/mdx';
   ```

```
import awsAmplify from 'astro-aws-amplify';

import sitemap from '@astrojs/sitemap';

// https://astro.build/config
export default defineConfig({
  site: 'https://example.com',
  integrations: [mdx(), sitemap()],
  adapter: awsAmplify(),
  output: 'server',
});
```

4. 변경 사항을 커밋하고 프로젝트를 Git 리포지토리로 푸시합니다.

   이제 Astro 앱을 Amplify에 배포할 준비가 되었습니다.

5. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
6. 모든 앱 페이지에서 새 앱 생성을 선택합니다.
7. Amplify로 빌드 시작 페이지에서 Git 리포지토리 공급자를 선택하고 다음을 선택합니다.
8. 리포지토리 브랜치 추가 페이지에서 다음을 수행합니다.

   a. 연결할 리포지토리의 이름을 선택합니다.

   b. 연결할 리포지토리 브랜치의 이름을 선택합니다.

   c. 다음을 선택합니다.
9. 앱 설정 페이지에서 빌드 설정 섹션을 찾습니다. 빌드 출력 디렉터리에 **.amplify-hosting**을 입력합니다.
10. 빌드 사양에서 앱의 프론트엔드 빌드 명령도 업데이트해야 합니다. 빌드 사양을 열려면 YML 파일 편집을 선택합니다.
11. `amplify.yml` 파일에서 프론트엔드 빌드 명령 섹션을 찾습니다. **mv node_modules ./.amplify-hosting/compute/default**를 입력합니다.

    빌드 설정 파일이 다음과 같아야 합니다.

```
version: 1
frontend:
    phases:
        preBuild:
            commands:
                - 'npm ci --cache .npm --prefer-offline'
        build:
```

```
            commands:
                - 'npm run build'
                - 'mv node_modules ./.amplify-hosting/compute/default'
    artifacts:
        baseDirectory: .amplify-hosting
        files:
            - '**/*'
    cache:
        paths:
            - '.npm/**/*'
```

12. 저장을 선택합니다.
13. Amplify가 Amazon CloudWatch Logs에 앱 로그를 전달할 수 있도록 하려면 콘솔에서 명시적으로 활성화해야 합니다. 고급 설정 섹션을 열고 서버 측 렌더링(SSR) 배포 섹션에서 SSR 앱 로그 활성화를 선택합니다.
14. 다음을 선택합니다.
15. 검토 페이지에서 저장 및 배포를 선택합니다.

Amplify Hosting에 SvelteKit 앱 배포

SvelteKit 애플리케이션을 Amplify Hosting에 배포하려면 다음 지침을 따릅니다. 자체 애플리케이션을 사용하거나 스타터 앱을 생성할 수 있습니다. 자세한 내용은 SvelteKit 설명서의 프로젝트 생성을 참조하세요.

SSR로 SvelteKit 앱을 Amplify Hosting에 배포하려면 프로젝트에 어댑터를 추가해야 합니다. 당사는 Amplify 소유의 SvelteKit 프레임워크용 어댑터를 유지 관리하지 않습니다. 이 예제에서는 커뮤니티의 구성원이 생성한 `amplify-adapter`를 사용합니다. 어댑터는 GitHub 웹 사이트의 github.com/gzimbron/amplify-adapter 사용할 수 있습니다.는이 어댑터를 유지 관리하지 AWS 않습니다.

Amplify Hosting에 SvelteKit 앱을 배포하려면

1. 로컬 컴퓨터에서 배포할 SvelteKit 애플리케이션으로 이동합니다.
2. 어댑터를 설치하려면 터미널 창을 열고 다음 명령을 실행합니다. 이 예제에서는 github.com/gzimbron/amplify-adapter에서 사용할 수 있는 커뮤니티 어댑터를 사용합니다. 다른 커뮤니티 어댑터를 사용하는 경우 *`amplify-adapter`*를 해당 어댑터의 이름으로 바꿉니다.

```
npm install amplify-adapter
```

3. SvelteKit 앱의 프로젝트 폴더에서 `svelte.config.js` 파일을 엽니다. `amplify-adapter`를 사용하도록 파일을 편집하거나 *`'amplify-adapter'`*를 사용할 어댑터의 이름으로 바꿉니다. 파일이 다음과 같아야 합니다.

```
import adapter from 'amplify-adapter';
import { vitePreprocess } from '@sveltejs/vite-plugin-svelte';


/** @type {import('@sveltejs/kit').Config} */
const config = {
        // Consult https://kit.svelte.dev/docs/integrations#preprocessors
        // for more information about preprocessors
        preprocess: vitePreprocess(),

        kit: {
                // adapter-auto only supports some environments, see https://
kit.svelte.dev/docs/adapter-auto for a list.
                // If your environment is not supported, or you settled on a
 specific environment, switch out the adapter.
                // See https://kit.svelte.dev/docs/adapters for more information
 about adapters.
                adapter: adapter()
        }
};

export default config;
```

4. 변경 사항을 커밋하고 애플리케이션을 Git 리포지토리로 푸시합니다.
5. 이제 SvelteKit 앱을 Amplify에 배포할 준비가 되었습니다.

   에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
6. 모든 앱 페이지에서 새 앱 생성을 선택합니다.
7. Amplify로 빌드 시작 페이지에서 Git 리포지토리 공급자를 선택하고 다음을 선택합니다.
8. 리포지토리 브랜치 추가 페이지에서 다음을 수행합니다.

   a. 연결할 리포지토리의 이름을 선택합니다.

   b. 연결할 리포지토리 브랜치의 이름을 선택합니다.

   c. 다음을 선택합니다.
9. 앱 설정 페이지에서 빌드 설정 섹션을 찾습니다. 빌드 출력 디렉터리에 **`build`**을 입력합니다.

10. 빌드 사양에서 앱의 프론트엔드 빌드 명령도 업데이트해야 합니다. 빌드 사양을 열려면 YML 파일 편집을 선택합니다.
11. amplify.yml 파일에서 프론트엔드 빌드 명령 섹션을 찾습니다. **- cd build/compute/default/** 및 **- npm i --production**을 입력합니다.

    빌드 설정 파일이 다음과 같아야 합니다.

```
version: 1
frontend:
    phases:
        preBuild:
            commands:
                - 'npm ci --cache .npm --prefer-offline'
        build:
            commands:
                - 'npm run build'
                - 'cd build/compute/default/'
                - 'npm i --production'

    artifacts:
        baseDirectory: build
        files:
            - '**/*'
    cache:
        paths:
            - '.npm/**/*'
```

12. 저장을 선택합니다.
13. Amplify가 Amazon CloudWatch Logs에 앱 로그를 전달할 수 있도록 하려면 콘솔에서 명시적으로 활성화해야 합니다. 고급 설정 섹션을 열고 서버 측 렌더링(SSR) 배포 섹션에서 SSR 앱 로그 활성화를 선택합니다.
14. 다음을 선택합니다.
15. 검토 페이지에서 저장 및 배포를 선택합니다.

Amplify Hosting을 통해 서버 측 렌더링 애플리케이션 배포

AWS Amplify 를 사용하여 서버 측 렌더링(SSR)을 사용하는 웹 앱을 배포하고 호스팅할 수 있습니다. Amplify Hosting에서는 Next.js 프레임워크를 사용하여 생성한 애플리케이션이 자동으로 감지되며 AWS Management Console에서 수동으로 구성할 필요가 없습니다.

Amplify에서는 애플리케이션의 빌드 출력을 Amplify Hosting에서 예상하는 디렉터리 구조로 변환하는 오픈 소스 빌드 어댑터가 있는 Javascript 기반 SSR 프레임워크도 지원됩니다. 예를 들어 사용 가능한 어댑터를 설치하여 Nuxt, Astro 및 SvelteKit 프레임워크로 생성된 앱을 배포할 수 있습니다.

고급 사용자는 배포 사양을 사용하여 빌드 어댑터를 생성하거나 빌드 후 스크립트를 구성할 수 있습니다.

최소한의 구성으로 다음 프레임워크를 Amplify Hosting에 배포할 수 있습니다.

Next.js

- Amplify는 어댑터 없이 Next.js 15 애플리케이션을 지원합니다. 시작하려면 Next.js에 대한 Amplify 지원 섹션을 참조하세요.

Nuxt.js

- Amplify는 사전 설정된 어댑터를 사용하여 Nuxt.js 애플리케이션 배포를 지원합니다. 시작하려면 Nuxt.js에 대한 Amplify 지원 섹션을 참조하세요.

Astro.js

- Amplify는 커뮤니티 어댑터를 사용하여 Astro.js 애플리케이션 배포를 지원합니다. 시작하려면 Astro.js에 대한 Amplify 지원 섹션을 참조하세요.

SvelteKit

- Amplify는 커뮤니티 어댑터를 사용하여 SvelteKit 애플리케이션 배포를 지원합니다. 시작하려면 SvelteKit에 대한 Amplify 지원 섹션을 참조하세요.

오픈 소스 어댑터

- 오픈 소스 어댑터 사용 - 이전 목록에 없는 어댑터 사용에 대한 지침은 모든 SSR 프레임워크에 오픈 소스 어댑터 사용 섹션을 참조하세요.
- 프레임워크 어댑터 구축 - 프레임워크 제공 기능을 통합하려는 프레임워크 작성자는 Amplify Hosting 배포 사양을 참조하여 Amplify에서 예상하는 구조에 부합하도록 빌드 출력을 구성할 수 있습니다. 자세한 내용은 Amplify Hosting 배포 사양을 참조하여 빌드 출력 구성을 참조하세요.

- 빌드 후 스크립트 구성 - Amplify Hosting 배포 사양을 참조하여 특정 시나리오에 필요한 대로 빌드 출력을 조작할 수 있습니다. 자세한 내용은 Amplify Hosting 배포 사양을 참조하여 빌드 출력 구성을 참조하세요. 예제는 배포 매니페스트를 사용하여 Express 서버 배포 섹션을 참조하세요.

주제

Next.js에 대한 Amplify 지원

Amplify는 Next.js를 사용하여 생성된 서버 측 렌더링(SSR) 웹 앱에 대한 배포 및 호스팅을 지원합니다. Next.js는 JavaScript로 SPA를 개발하기 위한 React 프레임워크입니다. 이미지 최적화 및 미들웨어와 같은 기능을 사용하여 Next.js 15까지 Next.js 버전으로 빌드된 앱을 배포할 수 있습니다.

개발자는 Next.js를 사용하여 정적 사이트 생성(SSG)과 SSR을 단일 프로젝트에 결합할 수 있습니다. SSG 페이지는 빌드 시 프리렌더링되며, SSR 페이지는 요청 시 프리렌더링됩니다.

프리렌더링은 성능과 검색 엔진 최적화를 개선할 수 있습니다. Next.js는 서버의 모든 페이지를 프리렌더링하므로 각 페이지의 HTML 콘텐츠가 클라이언트의 브라우저에 도달되는 즉시 이용할 수 있습니다. 이러한 콘텐츠는 또한 더 빨리 로드될 수 있습니다. 로드 시간이 빨라지면 최종 사용자의 웹사이트 이용 경험이 향상되고 사이트의 SEO 순위에 긍정적인 영향을 미칩니다. 또한 프리렌더링은 검색 엔진 봇이 웹사이트의 HTML 콘텐츠를 쉽게 찾고 크롤링할 수 있도록 하여 SEO를 개선합니다.

Next.js는 첫 번째 바이트까지의 시간(TTFB)과 첫 번째 콘텐츠를 렌더링하는 데 걸리는 시간(FCP)과 같은 다양한 성능 메트릭을 측정하기 위한 내장된 분석 지원을 제공합니다. Next.js에 관한 자세한 내용은 Next.js 웹사이트의 시작하기를 참조하십시오.

Next.js 기능 지원

Amplify Hosting 컴퓨팅은 Next.js 버전 12~15로 빌드된 앱의 서버 측 렌더링(SSR)을 완전히 관리합니다.

2022년 11월에 Amplify Hosting 컴퓨팅이 출시되기 전에 Amplify에 Next.js 앱을 배포한 경우 앱은 Amplify의 이전 SSR 공급자인 Classic(Next.js 11만 해당)을 사용하고 있습니다. Amplify Hosting 컴퓨팅은 Next.js 11 또는 그 이전 버전을 사용하여 만든 앱을 지원하지 않습니다. Next.js 11 앱을 Amplify Hosting 컴퓨팅 관리형 SSR 공급자로 마이그레이션하는 것이 좋습니다.

다음 목록은 Amplify Hosting 컴퓨팅 SSR 공급자가 지원하는 특정 기능을 설명합니다.

지원되는 기능

- 서버 측 렌더링 페이지(SSR)
- 정적 페이지
- API 라우팅
- 동적 라우팅
- 모든 라우팅 포착
- SSG(정적 생성)
- 증분 정적 재생성(ISR)
- 다국어(i18n) 하위 경로 라우팅
- 다국어(i18n) 도메인 라우팅
- 다국어(i18n) 자동 로케일 감지
- 미들웨어
- 환경 변수
- 이미지 최적화
- Next.js 13 앱 디렉터리

지원되지 않는 기능

- 엣지 API 경로(엣지 미들웨어 미지원)
- 온디맨드 증분 정적 재생성(ISR)
- Next.js 스트리밍

- 정적 자산 및 최적화된 이미지에서 미들웨어 실행
- 를 사용한 응답 후 코드 실행`unstable_after`(Experimental feature released with Next.js 15)

Next.js 이미지

이미지의 최대 출력 크기는 4.3MB를 초과할 수 없습니다. 더 큰 이미지 파일은 다른 곳에 저장해 두고 Next.js Image 구성 요소를 사용하여 Webp 또는 AVIF 형식으로 크기를 조정하고 최적화한 다음, 더 작은 크기로 제공할 수 있습니다.

참고로, Next.js 설명서에서는 프로덕션 환경에서 이미지 최적화가 올바르게 작동할 수 있도록 Sharp 이미지 처리 모듈을 설치할 것을 권장합니다. 그러나 Amplify 배포에는 필요하지 않습니다. Amplify는 Sharp를 자동으로 배포합니다.

Amplify에 Next.js SSR 애플리케이션 배포

기본적으로 Amplify는 Next.js 버전 12~15를 지원하는 Amplify Hosting의 컴퓨팅 서비스를 사용하여 새 SSR 앱을 배포합니다. Amplify Hosting 컴퓨팅에서는 SSR 앱을 배포하는 데 필요한 리소스를 완벽하게 관리합니다. 2022년 11월 17일 이전에 배포한 Amplify 계정의 SSR 앱은 Classic(Next.js 11만 해당) SSR 공급자를 사용합니다.

Classic(Next.js 11만 해당) SSR을 사용하는 앱을 Amplify Hosting 컴퓨팅 SSR 공급자로 마이그레이션하는 것이 좋습니다. Amplify는 자동 마이그레이션을 수행하지 않습니다. 앱을 수동으로 마이그레이션한 다음 새 빌드를 시작하여 업데이트를 완료해야 합니다. 지침은 Next.js 11 SSR 앱을 Amplify Hosting 컴퓨팅으로 마이그레이션 섹션을 참조하세요.

새 Next.js SSR 앱을 배포하려면 다음 지침을 따릅니다.

Amplify Hosting 컴퓨팅 SSR 공급자를 사용하여 Amplify에 SSR 앱을 배포하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 리포지토리 공급자를 선택하고 다음을 선택합니다.
4. 리포지토리 브랜치 추가 페이지에서 다음을 수행합니다.
    a. 최근 업데이트된 리포지토리 목록에서 연결할 리포지토리 이름을 선택합니다.
    b. 브랜치 목록에서 연결할 리포지토리 브랜치의 이름을 선택합니다.
    c. 다음을 선택합니다.

5. 앱은 Amplify가 사용자를 대신하여 다른 서비스를 호출할 때 맡는 IAM 서비스 역할을 필요로 합니다. Amplify Hosting 컴퓨팅이 자동으로 서비스 역할을 생성하도록 허용하거나 사용자가 생성한 역할을 지정할 수 있습니다.
   - Amplify가 자동으로 역할을 생성하여 앱에 연결할 수 있도록 하려면
     - 새 서비스 역할 생성 및 사용을 선택합니다.
   - 이전에 생성한 서비스 역할을 연결하려면
     a. 기존 서비스 역할 사용을 선택합니다.
     b. 목록에서 사용할 역할을 선택합니다.
6. 다음을 선택합니다.
7. 검토 페이지에서 저장 및 배포를 선택합니다.

Package.json 파일 설정

Next.js 앱 배포 시 Amplify는 `package.json` 파일에 있는 앱의 빌드 스크립트를 검사하여 애플리케이션 유형을 결정합니다.

다음은 Next.js 앱의 빌드 스크립트 예입니다. 빌드 스크립트 `"next build"`는 앱이 SSG 페이지와 SSR 페이지를 모두 지원함을 나타냅니다. 이 빌드 스크립트는 Next.js 14 이상 SSG 전용 앱에도 사용됩니다.

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start"
},
```

다음은 Next.js 13 또는 이전 SSG 앱의 빌드 스크립트 예입니다. 빌드 스크립트 `"next build && next export"`는 앱이 SSG 페이지만 지원함을 나타냅니다.

```
"scripts": {
  "dev": "next dev",
  "build": "next build && next export",
  "start": "next start"
},
```

Next.js SSR 애플리케이션의 Amplify 빌드 설정

Amplify는 앱의 `package.json` 파일을 검사한 후 앱의 빌드 설정을 확인합니다. Amplify 콘솔이나 리포지토리 루트의 `amplify.yml` 파일에 빌드 설정을 저장할 수 있습니다. 자세한 내용은 Amplify 애플리케이션의 빌드 설정 구성 섹션을 참조하십시오.

Next.js SSR 앱을 배포하고 있는 것을 감지했으나 `amplify.yml` 파일이 없는 경우, Amplify는 앱에 대한 buildspec을 생성하고 `baseDirectory`을(를) `.next`(으)로 설정합니다. `amplify.yml` 파일이 있는 곳에 앱을 배포하는 경우, 파일의 빌드 설정이 콘솔의 모든 빌드 설정을 재정의합니다. 따라서 파일에서 `baseDirectory`를 `.next`으로 수동 설정해야 합니다.

다음은 `baseDirectory`가 `.next`으로 설정된 앱의 빌드 설정 예시입니다. 이는 빌드 아티팩트가 SSG 및 SSR 페이지를 지원하는 Next.js 앱용임을 나타냅니다.

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

Next.js 13 또는 이전 SSG 애플리케이션의 Amplify 빌드 설정

Amplify는 Next.js 13 또는 이전 SSG 앱이 배포되는 것을 감지하면 앱에 대한 빌드 사양을 생성하고 `baseDirectory`를 `out`으로 설정합니다. `amplify.yml` 파일이 있는 곳에 앱을 배포하는 경우, 파일에서 `baseDirectory`을(를) `out`(으)로 수동으로 설정해야 합니다. `out` 디렉터리는 Next.js가 내보낸 정적 자산을 저장하기 위해 생성하는 기본 폴더입니다. 앱의 빌드 사양 설정을 구성할 때 앱의 구성과 일치하도록 `baseDirectory` 폴더의 이름을 변경합니다.

다음은 빌드 아티팩트가 SSG 페이지만 지원하는 Next.js 13 또는 이전 앱에 대한 것임을 나타내도록 `baseDirectory`가 `out`으로 설정된 앱 빌드 설정의 예입니다.

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: out
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

Next.js 14 이상 SSG 애플리케이션에 대한 Amplify 빌드 설정

Next.js 버전 14에서는 `next export` 명령이 더 이상 사용되지 않고 `next.config.js` 파일에서 `output: 'export'`로 대체되어 정적 내보내기를 활성화합니다. 콘솔에서 Next.js 14 SSG 전용 애플리케이션을 배포하는 경우 Amplify는 앱의 빌드 사양을 생성하고 `baseDirectory`을 `.next`로 설정합니다. `amplify.yml` 파일이 있는 곳에 앱을 배포하는 경우, 파일에서 `baseDirectory`을(를) `.next`(으)로 수동으로 설정해야 합니다. 이는 Amplify가 SSG 및 SSR 페이지를 모두 지원하는 Next.js `WEB_COMPUTE` 애플리케이션에 사용하는 것과 동일한 `baseDirectory` 설정입니다.

다음은 `baseDirectory`가 `.next`로 설정된 Next.js 14 SSG 전용 앱 빌드 설정의 예입니다.

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
```

```
    paths:
      - node_modules/**/*
```

Next.js 11 SSR 앱을 Amplify Hosting 컴퓨팅으로 마이그레이션

새 Next.js 앱 배포 시 Amplify는 지원되는 가장 최신 버전의 Next.js를 기본으로 사용합니다. 현재 Amplify Hosting 컴퓨팅 SSR 공급자는 Next.js 버전 15를 지원합니다.

Amplify 콘솔은 Next.js 버전 12~15를 완전히 지원하는 Amplify Hosting 컴퓨팅 서비스의 2022년 11월 릴리스 이전에 배포된 계정의 앱을 감지합니다. 이 콘솔은 Amplify의 이전 SSR 공급자인 Classic(Next.js 11만 해당)을 사용하여 배포된 브랜치가 있는 앱을 식별하는 정보 배너를 보여줍니다. Amplify Hosting 컴퓨팅 SSR 공급자로 앱을 마이그레이션하는 것이 좋습니다.

호스팅된 Next.js 11 애플리케이션을 Next.js 12 이상으로 업데이트하는 경우 배포가 트리거될 때 `"target" property is no longer supported` 오류가 발생할 수 있습니다. 이 경우 Amplify Hosting 컴퓨팅으로 마이그레이션해야 합니다.

앱과 모든 프로덕션 브랜치를 동시에 수동으로 마이그레이션해야 합니다. 앱은 Classic(Next.js 11만 해당)과 Next.js 12 이상 브랜치를 둘 다 포함할 수 없습니다.

다음 지침에 따라 앱을 Amplify Hosting 컴퓨팅 SSR 공급자로 마이그레이션합니다.

앱을 Amplify Hosting 컴퓨팅 SSR 공급자로 마이그레이션하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 마이그레이션하려는 Next.js 앱을 선택합니다.

> **Note**
>
> Amplify 콘솔에서 앱을 마이그레이션하기 전에 Next.js 12 또는 그 이후 버전을 사용할 수 있도록 먼저 앱의 package.json 파일을 업데이트해야 합니다.

3. 탐색 창에서 앱 설정, 일반을 선택합니다.
4. 앱에 Classic(Next.js 11만 해당) SSR 공급자를 사용하여 배포한 브랜치가 있는 경우, 앱 홈페이지의 콘솔에 배너가 표시됩니다. 배너에서 마이그레이션을 선택합니다.
5. 마이그레이션 확인 창에서 세 개의 명령문을 선택하고 마이그레이션을 선택합니다.
6. Amplify가 앱을 빌드하고 재배포하여 마이그레이션을 완료할 것입니다.

SSR 마이그레이션 되돌리기

Next.js 앱 배포 시 Amplify Hosting은 앱의 설정을 감지하고 앱의 내부 플랫폼 값을 설정합니다. 세 개의 유효한 플랫폼 값이 있습니다. SSG 앱의 플랫폼 값은 WEB으로 설정됩니다. Next.js 버전 11을 사용하는 SSR 앱의 플랫폼 값은 WEB_DYNAMIC으로 설정됩니다. Next.js 12 또는 그 이후 버전 SSR 앱의 플랫폼 값은 WEB_COMPUTE으로 설정됩니다.

이전 섹션의 지침에 따라 앱을 마이그레이션하면 Amplify는 앱의 플랫폼 값을 WEB_DYNAMIC에서 WEB_COMPUTE으로 변경합니다. Amplify Hosting 컴퓨팅으로 마이그레이션을 완료한 후에는 콘솔에서 마이그레이션을 되돌릴 수 없습니다. 마이그레이션을 되돌리려면 AWS Command Line Interface 을 사용하여 앱의 플랫폼을 WEB_DYNAMIC으로 다시 변경해야 합니다. 터미널 창을 열고 다음 명령을 입력하여 사용자의 고유 정보로 앱 ID와 리전을 업데이트합니다.

```
aws amplify update-app --app-id abcd1234 --platform WEB_DYNAMIC --region us-west-2
```

정적 Next.js 앱에 SSR 기능 추가

Amplify를 통해 배포된 기존 정적(SSG) Next.js 앱에 SSR 기능을 추가할 수 있습니다. SSG 앱을 SSR로 변환하는 프로세스를 시작하기 전에, Next.js 12 또는 그 이후 버전을 사용하도록 앱을 업데이트하고 SSR 기능을 추가합니다. 그런 다음 다음 단계를 수행해야 합니다.

1. AWS Command Line Interface 를 사용하여 앱의 플랫폼 유형을 변경합니다.
2. 앱에 서비스 역할을 추가합니다.
3. 앱의 빌드 설정에서 출력 디렉터리를 업데이트합니다.
4. 앱이 SSR을 사용한다는 것을 나타내도록 앱의 package.json 파일을 업데이트합니다.

플랫폼 업데이트

플랫폼 유형에는 세 가지 유효한 값이 있습니다. SSG 앱의 플랫폼 유형은 WEB으로 설정됩니다. Next.js 버전 11을 사용하는 SSR 앱의 플랫폼 유형은 WEB_DYNAMIC으로 설정됩니다. Amplify Hosting 컴퓨팅이 관리하는 SSR을 사용하여 Next.js 12 이상에 배포된 앱의 경우, 플랫폼 유형이 WEB_COMPUTE으로 설정됩니다.

앱을 SSG 앱으로 배포한 경우, Amplify는 플랫폼 유형을 WEB으로 설정합니다. AWS CLI 를 사용하여 앱의 플랫폼을 로 변경합니다WEB_COMPUTE. 터미널 창을 열고 다음 명령을 입력하여 빨간색 텍스트를 사용자의 고유한 앱 ID 및 리전으로 업데이트합니다.

```
aws amplify update-app --app-id abcd1234 --platform WEB_COMPUTE --region us-west-2
```

서비스 역할 추가

서비스 역할은 Amplify가 사용자를 대신하여 다른 서비스를 호출할 때 수임하는 AWS Identity and Access Management (IAM) 역할입니다. Amplify를 통해 이미 배포된 SSG 앱에 서비스 역할을 추가하려면 다음 단계를 따릅니다.

서비스 역할을 추가하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. Amplify 계정에 아직 서비스 역할을 생성하지 않은 경우 서비스 역할 추가를 참조하여 이 필수 단계를 완료합니다.
3. 서비스 역할을 추가할 정적 Next.js 앱을 선택합니다.
4. 탐색 창에서 앱 설정, 일반을 선택합니다.
5. 앱 세부 정보 페이지에서 편집을 선택합니다.
6. 서비스 역할에서 기존 서비스 역할의 이름 또는 2단계에서 만든 서비스 역할의 이름을 선택합니다.
7. 저장을 선택합니다.

빌드 설정 업데이트

SSR 기능을 사용하여 앱을 재배포하기 전에 출력 디렉토리가 `.next`으로 설정되도록 앱의 빌드 설정을 업데이트해야 합니다. Amplify 콘솔 또는 리포지토리에 저장된 `amplify.yml` 파일에서 빌드 설정을 편집할 수 있습니다. 자세한 내용은 Amplify 애플리케이션의 빌드 설정 구성 섹션을 참조하십시오.

다음은 `baseDirectory`가 `.next`으로 설정된 앱의 빌드 설정 예시입니다.

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
```

```
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

package.json 파일 업데이트

서비스 역할을 추가하고 빌드 설정을 업데이트한 후 앱의 `package.json` 파일을 업데이트합니다. 다음 예제와 같이 Next.js 앱이 SSG 및 SSR 페이지를 모두 지원한다는 것을 나타내도록 빌드 스크립트를 `"next build"`로 설정합니다.

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start"
},
```

Amplify는 리포지토리의 `package.json` 파일 변경을 감지하고 SSR 기능을 사용하여 앱을 재배포합니다.

환경 변수가 서버 측 런타임에 액세스할 수 있도록 만들기

Amplify Hosting은 Amplify 콘솔의 프로젝트 구성에 환경 변수를 설정하여 애플리케이션 빌드에 환경 변수를 추가할 수 있도록 지원합니다.

하지만 Next.js 서버 구성 요소는 기본적으로 이러한 환경 변수에 액세스할 수 없습니다. 이는 애플리케이션이 빌드 단계에서 사용하는 환경 변수에 저장된 모든 암호를 보호하기 위한 것입니다.

특정 환경 변수가 Next.js에 액세스할 수 있도록 하려면 Next.js가 인식하는 환경 파일에 해당 변수를 설정하도록 Amplify 빌드 사양 파일을 수정하면 됩니다. 이를 통해 Amplify는 애플리케이션을 빌드하기 전에 이러한 환경 변수를 로드할 수 있습니다.

> **Important**
>
> 배포 아티팩트에 액세스할 수 있는 사용자는 읽을 수 있으므로 환경 변수에 자격 증명, 보안 암호 또는 민감한 정보를 저장하지 않는 것이 좋습니다.
> SSR 컴퓨팅 함수에 AWS 리소스에 대한 액세스 권한을 부여하려면 IAM 역할을 사용하는 것이 좋습니다.

다음 빌드 사양 예제는 빌드 명령 섹션에 환경 변수를 추가하는 방법을 보여줍니다.

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - env | grep -e API_BASE_URL >> .env.production
        - env | grep -e NEXT_PUBLIC_ >> .env.production
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
      - .next/cache/**/*
```

이 예제의 빌드 명령 섹션에는 애플리케이션 빌드가 실행되기 전에 `.env.production` 파일에 환경 변수를 쓰는 두 개의 명령이 포함되어 있습니다. Amplify Hosting은 애플리케이션이 트래픽을 수신할 때 애플리케이션이 이러한 변수에 액세스할 수 있도록 합니다.

이전 예제에서 빌드 명령 섹션의 다음 줄은 빌드 환경에서 특정 변수를 가져와 `.env.production` 파일에 추가하는 방법을 설명합니다.

```
- env | grep -e API_BASE_URL -e APP_ENV >> .env.production
```

변수가 빌드 환경에 있는 경우, `.env.production` 파일에는 다음과 같은 환경 변수가 포함됩니다.

```
API_BASE_URL=localhost
APP_ENV=dev
```

이전 예제에서 빌드 명령 섹션의 다음 줄은 특정 접두사가 있는 환경 변수를 `.env.production` 파일에 추가하는 방법을 설명합니다. 이 예제에서는 접두사 `NEXT_PUBLIC_`이 있는 모든 변수를 추가합니다.

```
- env | grep -e NEXT_PUBLIC_ >> .env.production
```

빌드 환경에 접두사 NEXT_PUBLIC_이 있는 변수가 여러 개 있는 경우 .env.production 파일은 다음과 유사하게 표시됩니다.

```
NEXT_PUBLIC_ANALYTICS_ID=abcdefghijk
NEXT_PUBLIC_GRAPHQL_ENDPOINT=uowelalsmlsadf
NEXT_PUBLIC_FEATURE_FLAG=true
```

모노 리포지토리용 SSR 환경 변수

모노 리포지토리에 SSR 앱을 배포하고 특정 환경 변수가 Next.js에 액세스할 수 있도록 하려면 .env.production 파일에 애플리케이션 루트 접두사를 붙여야 합니다. Nx 모노 리포지토리 내의 Next.js 앱에 대한 다음 빌드 사양 예제는 빌드 명령 섹션에 환경 변수를 추가하는 방법을 보여줍니다.

```
version: 1
applications:
  - frontend:
      phases:
        preBuild:
          commands:
            - npm ci
        build:
          commands:
            - env | grep -e API_BASE_URL -e APP_ENV >> apps/app/.env.production
            - env | grep -e NEXT_PUBLIC_ >> apps/app/.env.production
            - npx nx build app
      artifacts:
        baseDirectory: dist/apps/app/.next
        files:
          - '**/*'
      cache:
        paths:
          - node_modules/**/*
      buildPath: /
    appRoot: apps/app
```

이전 예제의 빌드 명령 섹션에 있는 다음 줄은 빌드 환경에서 특정 변수를 가져와 모노 리포지토리 내의 앱에 대한 .env.production 파일(애플리케이션 루트 apps/app)에 추가하는 방법을 설명합니다.

```
- env | grep -e API_BASE_URL -e APP_ENV >> apps/app/.env.production
- env | grep -e NEXT_PUBLIC_ >> apps/app/.env.production
```

모노레포 Next.js 앱 배포

Amplify는 일반 모노레포 앱뿐만 아니라 npm 워크스페이스, pnpm 워크스페이스, Yarn 워크스페이스, Nx 및 Turborepo를 사용하여 생성된 모노레포 앱을 지원합니다. 앱 배포 시 Amplify는 사용 중인 모노레포 빌드 프레임워크를 자동으로 감지합니다. Amplify는 npm 워크스페이스, Yarn 워크스페이스 또는 Nx의 앱에 대한 빌드 설정을 자동으로 적용합니다. Turborepo와 pnpm 앱에는 추가 구성이 필요합니다. 자세한 내용은 모노 리포지토리 빌드 설정 구성 단원을 참조하십시오.

Nx에 대한 자세한 예제는 AWS Amplify Hosting을 통해 Nx를 사용하는 Next.js 앱 간 코드 공유 블로그 게시물을 참조하십시오.

Nuxt.js에 대한 Amplify 지원

Nuxt는 Vue.js를 사용하여 풀 스택 웹 애플리케이션을 생성하기 위한 프레임워크입니다.

어댑터

사전 설정된 어댑터를 사용하여 제로 구성으로 Nuxt.js 애플리케이션을 Amplify에 배포할 수 있습니다. 어댑터에 대한 자세한 내용은 Nuxt 설명서를 참조하세요.

자습서

Nuxt.js 앱을 Amplify에 배포하는 방법은 Amplify Hosting에 Nuxt.js 앱 배포 섹션을 참조하세요.

데모

동영상 데모는 YouTube의 Nuxt Hosting with ZERO Configuration In Minutes(With AWS)를 참조하세요.

Astro.js에 대한 Amplify 지원

Astro는 콘텐츠 기반 웹 애플리케이션을 생성하기 위한 웹 프레임워크입니다.

어댑터

커뮤니티 어댑터를 사용하여 Astro.js 애플리케이션을 Amplify에 배포할 수 있습니다. 당사는 Amplify 소유의 Astro 프레임워크용 어댑터를 유지 관리하지 않습니다. 그러나 어댑터는 GitHub 웹 사이트의 github.com/alexnguyennz/astro-aws-amplify에서 사용할 수 있습니다. 이 어댑터는 커뮤니티의 구성원이 만들었으며 AWS가 유지 관리하지 않습니다.

자습서

Amplify에 Astro 앱을 배포하는 방법은 Amplify Hosting에 Astro.js 앱 배포 섹션을 참조하세요.

데모

동영상 데모는 Amazon Web Services YouTube 채널에서 How to deploy an Astro Website to AWS를 시청하세요.

SvelteKit에 대한 Amplify 지원

SvelteKit는 Svelte를 사용하여 풀 스택 웹 애플리케이션을 생성하기 위한 프레임워크입니다.

어댑터

커뮤니티 어댑터를 사용하여 SvelteKit 애플리케이션을 Amplify에 배포할 수 있습니다. 당사는 Amplify 소유의 SvelteKit 프레임워크용 어댑터를 유지 관리하지 않습니다. 하지만 어댑터는 GitHub 웹 사이트의 github.com/gzimbron/amplify-adapter 사용할 수 있습니다. 이 어댑터는 커뮤니티의 구성원이 만들었으며 AWS가 유지 관리하지 않습니다.

자습서

SvelteKit 앱을 Amplify에 배포하는 방법은 Amplify Hosting에 SvelteKit 앱 배포 섹션을 참조하세요.

데모

동영상 데모는 Amazon Web Services YouTube 채널에서 How to deploy a SvelteKit website (with API) to AWS를 시청하세요.

Amplify에 SSR 앱 배포

다음 지침을 참조하여 Amplify에서 예상하는 빌드 출력에 부합하는 배포 번들과 모든 프레임워크로 만든 앱을 배포할 수 있습니다. Next.js 애플리케이션을 배포하는 경우 어댑터가 필요하지 않습니다.

프레임워크 어댑터가 사용되는 SSR 앱을 배포한다면 먼저 어댑터를 설치하고 구성해야 합니다. 지침은 모든 SSR 프레임워크에 오픈 소스 어댑터 사용 섹션을 참조하세요.

Amplify Hosting에 SSR 앱을 배포하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.

2. 모든 앱 페이지에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 리포지토리 공급자를 선택하고 다음을 선택합니다.
4. 리포지토리 브랜치 추가 페이지에서 다음을 수행하십시오.

   a. 연결할 리포지토리의 이름을 선택합니다.
   b. 연결할 리포지토리 브랜치의 이름을 선택합니다.
   c. 다음을 선택합니다.
5. 앱 설정 페이지에서 Amplify는 Next.js SSR 앱을 자동으로 감지합니다.

   다른 프레임워크용 어댑터가 사용되는 SSR 앱을 배포하는 경우 Amazon CloudWatch Logs를 명시적으로 활성화해야 합니다. 고급 설정 섹션을 열고 서버 측 렌더링(SSR) 배포 섹션에서 SSR 앱 로그 활성화를 선택합니다.
6. Amplify가 AWS 계정으로 로그를 전달하는 IAM 서비스 역할이 앱에 필요합니다.

   서비스 역할을 추가하는 절차는 새 역할을 생성할지 아니면 기존 역할을 사용할지에 따라 달라집니다.
   - 새 역할을 생성하려면
     - 새 서비스 역할 생성 및 사용을 선택합니다.
   - 기존 역할을 사용하려면

     a. 기존 역할 사용을 선택합니다.
     b. 서비스 역할 목록에서 사용할 역할을 선택합니다.
7. 다음을 선택합니다.
8. 검토 페이지에서 저장 및 배포를 선택합니다.

SSR에 대해 지원되는 기능

이 섹션에서는 Amplify의 SSR 기능 지원에 대한 정보를 제공합니다.

Amplify는 앱을 빌드하는 데 사용된 Node.js 버전과 일치하도록 Node.js 버전 지원을 제공합니다.

Amplify에는 모든 SSR 앱을 지원하도록 기본적으로 제공되는 이미지 최적화 기능이 있습니다. 기본 이미지 최적화 기능을 사용하지 않으려면 사용자 지정 이미지 최적화 로더를 구현할 수 있습니다.

주제

- Next.js 앱에 대한 Node.js 버전 지원

- SSR 앱을 위한 이미지 최적화
- SSR 앱의 Amazon CloudWatch Logs
- Amplify Next.js 11 SSR 지원

Next.js 앱에 대한 Node.js 버전 지원

Amplify에서는 Next.js 컴퓨팅 앱을 빌드하고 배포할 때 앱 빌드에 사용된 Node.js의 메이저 버전과 일치하는 Node.js 런타임 버전을 사용합니다.

Amplify 콘솔의 라이브 패키지 재정의 기능에서 사용할 Node.js 버전을 지정할 수 있습니다. 라이브 패키지 업데이트 구성에 대한 자세한 내용은 빌드 이미지에서 특정 패키지 및 종속성 버전 사용 섹션을 참조하세요. nvm 명령과 같은 다른 메커니즘을 사용하여 Node.js 버전을 지정할 수도 있습니다. 버전을 지정하지 않으면 Amplify에서는 Amplify 빌드 컨테이너에서 사용되는 현재 버전을 기본적으로 사용합니다.

SSR 앱을 위한 이미지 최적화

Amplify Hosting에는 모든 SSR 앱을 지원하도록 기본적으로 제공되는 이미지 최적화 기능이 있습니다. Amplify의 이미지 최적화를 통해 가능한 한 가장 작은 파일 크기를 유지하면서 해당 이미지에 액세스하는 디바이스에 적합한 형식, 크기 및 해상도로 고품질 이미지를 전달할 수 있습니다.

현재 Next.js Image 구성 요소를 사용하여 온디맨드 방식으로 이미지를 최적화하거나 사용자 지정 이미지 로더를 구현할 수 있습니다. Next.js 13 이상을 사용한다면 Amplify의 이미지 최적화 기능을 사용하기 위해 별도의 조치를 취하지 않아도 됩니다. 사용자 지정 로더를 구현하는 경우 이어지는 사용자 지정 이미지 로더 사용을 참조하세요.

사용자 지정 이미지 로더 사용

사용자 지정 이미지 로더를 사용하면 Amplify에서는 애플리케이션의 `next.config.js` 파일에서 로더를 감지하며 기본으로 제공되는 이미지 최적화 기능을 활용하지 않습니다. Next.js에서 지원하는 사용자 지정 로더에 대한 자세한 내용은 Next.js 이미지 설명서를 참조하세요.

SSR 앱의 Amazon CloudWatch Logs

Amplify는 SSR 런타임에 대한 정보를 사용자 AWS 계정의 Amazon CloudWatch Logs로 보냅니다. SSR 앱을 배포할 때 Amplify가 사용자를 대신하여 다른 서비스를 호출할 때 맡는 IAM 서비스 역할이 앱에 필요합니다. Amplify Hosting 컴퓨팅이 자동으로 서비스 역할을 생성하도록 허용하거나 사용자가 생성한 역할을 지정할 수 있습니다.

Amplify에서 IAM 역할을 생성하도록 허용하면 해당 역할에 CloudWatch Logs를 생성할 권한이 부여됩니다. IAM 역할을 직접 생성하는 경우, Amplify가 Amazon CloudWatch Logs에 액세스할 수 있도록 하려면 정책에 다음 권한을 추가해야 합니다.

```
logs:CreateLogStream
logs:CreateLogGroup
logs:DescribeLogGroups
logs:PutLogEvents
```

서비스 역할에 대한 자세한 내용은 백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가을 참조하십시오.

Amplify Next.js 11 SSR 지원

2022년 11월 17일에 Amplify Hosting 컴퓨팅이 릴리스되기 전에 Amplify에 Next.js 앱을 배포한 경우, 앱은 Amplify의 이전 SSR 공급자인 Classic(Next.js 11만 해당)을 사용합니다. 이 섹션의 설명서는 Classic(Next.js 11만 해당) SSR 공급자를 사용하여 배포한 앱에만 적용됩니다.

> **Note**
>
> Next.js 11 앱을 Amplify Hosting 컴퓨팅 관리형 SSR 공급자로 마이그레이션하는 것이 좋습니다. 자세한 내용은 Next.js 11 SSR 앱을 Amplify Hosting 컴퓨팅으로 마이그레이션 단원을 참조하십시오.

다음 목록은 Amplify Classic(Next.js 11만 해당) SSR 공급자가 지원하는 특정 기능을 설명합니다.

지원되는 기능

- 서버 측 렌더링 페이지(SSR)
- 정적 페이지
- API 라우팅
- 동적 라우팅
- 모든 라우팅 포착
- SSG(정적 생성)
- 증분 정적 재생성(ISR)
- 다국어(i18n) 하위 경로 라우팅
- 환경 변수

지원되지 않는 기능

- 이미지 최적화
- 온디맨드 증분 정적 재생성(ISR)
- 다국어(i18n) 도메인 라우팅
- 다국어(i18n) 자동 로케일 감지
- 미들웨어
- 엣지 미들웨어
- 엣지 API 라우팅

Next.js 11 SSR 앱 요금

Next.js 11 SSR 앱을 배포할 때 Amplify는 AWS 계정에 다음을 포함한 추가 백엔드 리소스를 생성합니다.

- 앱의 정적 자산에 대한 리소스를 저장하는 Amazon Simple Storage Service(S3) 버킷. Amazon S3 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.
- 앱을 제공하기 위한 Amazon CloudFront 배포. CloudFront 요금에 대한 자세한 내용은 Amazon CloudFront 요금을 참조하십시오.
- CloudFront를 통해 전달되는 콘텐츠를 사용자 지정할 수 있는 네 가지 Lambda@Edge 기능입니다.

AWS Identity and Access Management Next.js 11 SSR 앱에 대한 권한

Amplify는 SSR 앱을 배포하려면 AWS Identity and Access Management (IAM) 권한이 필요합니다. SSR 앱의 경우 Amplify는 Amazon S3 버킷, CloudFront 배포, Lambda@Edge 함수, Amazon SQS 대기열(IAS를 사용하는 경우) 및 IAM 역할과 같은 리소스를 배포합니다. 필요한 최소 권한이 없으면 SSR 앱을 배포하려고 할 때 `Access Denied` 오류가 발생합니다. Amplify에 필요한 권한을 제공하려면 서비스 역할을 지정해야 합니다.

Amplify가 사용자를 대신하여 다른 서비스를 호출할 때 맡는 IAM 서비스 역할을 생성하려면 백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가 섹션을 참조하십시오. 이 지침은 `AdministratorAccess-Amplify` 관리형 정책을 연결하는 역할을 생성하는 방법을 설명합니다.

`AdministratorAccess-Amplify` 관리형 정책은 IAM 작업을 포함한 여러 AWS 서비스에 대한 액세스를 제공합니다. 및는 `AdministratorAccess` 정책만큼 강력한 것으로 간주되어야 합니다. 이 정책은 SSR 앱을 배포하는 데 필요한 것보다 더 많은 권한을 제공합니다.

최소 권한을 부여하는 모범 사례를 따르고 서비스 역할에 부여되는 권한을 줄이는 것이 좋습니다. 관리자에게 서비스 역할에 대한 액세스 권한을 부여하는 대신, SSR 앱을 배포하는 데 필요한 권한만 부여하는 자체 고객 관리형 IAM 정책을 만들 수 있습니다. 고객 관리형 정책을 만드는 방법에 대한 지침은 IAM 사용 설명서의 IAM 정책 생성 섹션을 참조하십시오.

자체 정책을 생성하는 경우, SSR 앱을 배포하는 데 필요한 다음의 최소 권한 목록을 참조하십시오.

```
acm:DescribeCertificate
acm:DescribeCertificate
acm:ListCertificates
acm:RequestCertificate
cloudfront:CreateCloudFrontOriginAccessIdentity
cloudfront:CreateDistribution
cloudfront:CreateInvalidation
cloudfront:GetDistribution
cloudfront:GetDistributionConfig
cloudfront:ListCloudFrontOriginAccessIdentities
cloudfront:ListDistributions
cloudfront:ListDistributionsByLambdaFunction
cloudfront:ListDistributionsByWebACLId
cloudfront:ListFieldLevelEncryptionConfigs
cloudfront:ListFieldLevelEncryptionProfiles
cloudfront:ListInvalidations
cloudfront:ListPublicKeys
cloudfront:ListStreamingDistributions
cloudfront:UpdateDistribution
cloudfront:TagResource
cloudfront:UntagResource
cloudfront:ListTagsForResource
iam:AttachRolePolicy
iam:CreateRole
iam:CreateServiceLinkedRole
iam:GetRole
iam:PutRolePolicy
iam:PassRole
lambda:CreateFunction
lambda:EnableReplication
lambda:DeleteFunction
lambda:GetFunction
lambda:GetFunctionConfiguration
lambda:PublishVersion
lambda:UpdateFunctionCode
lambda:UpdateFunctionConfiguration
```

```
lambda:ListTags
lambda:TagResource
lambda:UntagResource
route53:ChangeResourceRecordSets
route53:ListHostedZonesByName
route53:ListResourceRecordSets
s3:CreateBucket
s3:GetAccelerateConfiguration
s3:GetObject
s3:ListBucket
s3:PutAccelerateConfiguration
s3:PutBucketPolicy
s3:PutObject
s3:PutBucketTagging
s3:GetBucketTagging
lambda:ListEventSourceMappings
lambda:CreateEventSourceMapping
iam:UpdateAssumeRolePolicy
iam:DeleteRolePolicy
sqs:CreateQueue           // SQS only needed if using ISR feature
sqs:DeleteQueue
sqs:GetQueueAttributes
sqs:SetQueueAttributes
amplify:GetApp
amplify:GetBranch
amplify:UpdateApp
amplify:UpdateBranch
```

Next.js 11 SSR 배포 문제 해결

Amplify로 Classic(Next.js 11만 해당) SSR 앱을 배포할 때 예상치 못한 문제가 발생하는 경우, 다음 문제 해결 항목을 검토하십시오.

주제

- 애플리케이션의 출력 디렉터리가 재정의됨
- SSR 사이트를 배포한 후 404 오류가 발생함
- 애플리케이션에 CloudFront SSR 배포에 대한 다시 쓰기 규칙이 누락되어 있음
- 애플리케이션이 너무 커서 배포할 수 없음
- 메모리 부족 오류로 인해 빌드가 실패함
- 애플리케이션에 SSR 브랜치와 SSG 브랜치가 모두 있음

- 애플리케이션이 예약된 경로의 폴더에 정적 파일을 저장 중임
- 애플리케이션이 CloudFront 한도에 도달함
- Lambda@Edge 함수가 미국 동부(버지니아 북부) 리전에서 생성됨
- Next.js 애플리케이션이 지원되지 않는 기능을 사용함
- Next.js 애플리케이션의 이미지가 로드되지 않음
- 지원되지 않는 리전

애플리케이션의 출력 디렉터리가 재정의됨

Amplify로 배포된 Next.js 앱의 출력 디렉토리는 `.next`으로 설정되어야 합니다. 앱의 출력 디렉토리가 재정의되는 경우 `next.config.js` 파일을 확인합니다. 빌드 출력 디렉터리의 기본값을 `.next`으로 설정하려면 파일에서 다음 줄을 삭제합니다.

```
distDir: 'build'
```

빌드 설정에서 출력 디렉터리가 `.next`으로 설정되어 있는지 확인합니다. 앱의 빌드 설정을 보는 방법에 대한 자세한 내용은 Amplify 애플리케이션의 빌드 설정 구성 섹션을 참조하십시오.

다음은 `baseDirectory`가 `.next`으로 설정된 앱의 빌드 설정 예시입니다.

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

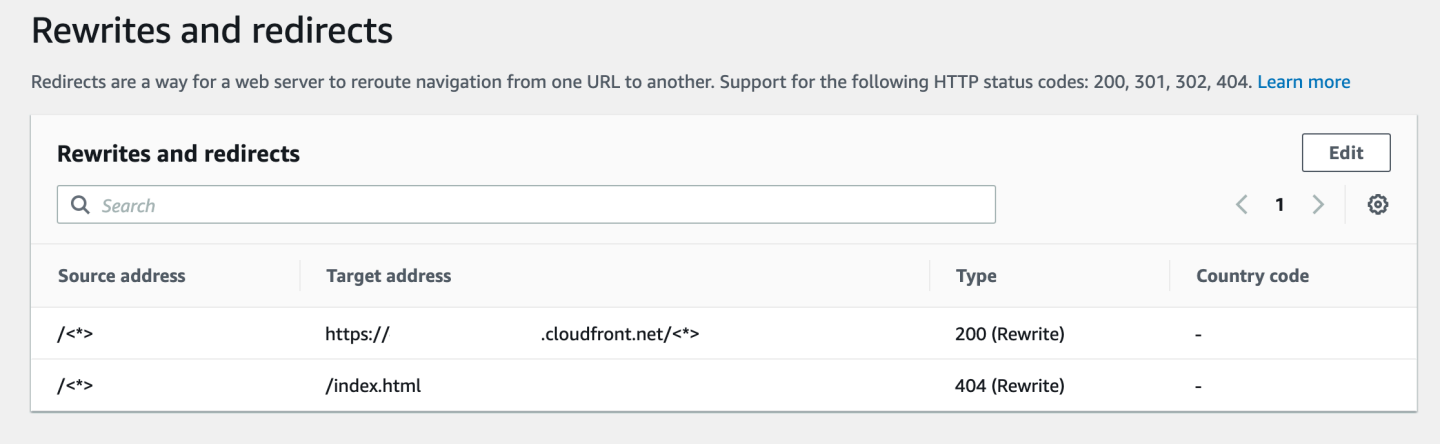
SSR 사이트를 배포한 후 404 오류가 발생함

사이트를 배포한 후 404 오류가 발생하는 경우, 출력 디렉터리가 재정의되어 문제가 발생한 것일 수 있습니다. `next.config.js` 파일을 확인하고 앱 빌드 사양의 빌드 출력 디렉터리가 올바른지 확인하려면 이전 애플리케이션의 출력 디렉터리가 재정의됨 주제의 단계를 따릅니다.

애플리케이션에 CloudFront SSR 배포에 대한 다시 쓰기 규칙이 누락되어 있음

SSR 앱 배포 시 Amplify는 CloudFront SSR 배포를 위한 다시 쓰기 규칙을 생성합니다. 웹 브라우저에서 앱에 액세스할 수 없는 경우, Amplify 콘솔에 앱에 대한 CloudFront 다시 쓰기 규칙이 있는지 확인합니다. 규칙이 없는 경우 수동으로 추가하거나 앱을 재배포할 수 있습니다.

Amplify 콘솔에서 앱의 다시 쓰기 및 리디렉션 규칙을 보거나 편집하려면, 탐색 창에서 앱 설정을 선택한 다음 다시 쓰기 및 리디렉션을 선택합니다. 다음 스크린샷은 SSR 앱을 배포할 때 Amplify가 생성하는 다시 쓰기 규칙의 예시를 보여줍니다. 이 예시에서는 CloudFront 다시 쓰기 규칙이 존재한다는 것을 알 수 있습니다.



애플리케이션이 너무 커서 배포할 수 없음

Amplify는 SSR 배포 크기를 50MB로 제한합니다. Next.js SSR 앱을 Amplify에 배포하려고 할 때 `RequestEntityTooLargeException` 오류가 발생한다면 앱이 너무 커서 배포할 수 없다는 의미입니다. `next.config.js` 파일에 캐시 정리 코드를 추가하여 이 문제를 해결할 수 있습니다.

다음은 캐시 정리를 수행하는 `next.config.js` 파일 내 코드의 예입니다.

```
module.exports = {
    webpack: (config, { buildId, dev, isServer, defaultLoaders, webpack }) => {
        config.optimization.splitChunks.cacheGroups = { }
        config.optimization.minimize = true;
        return config
```

```
      },
}
```

메모리 부족 오류로 인해 빌드가 실패함

Next.js를 통해 빌드 아티팩트를 캐시하여 후속 빌드의 성능을 개선할 수 있습니다. 또한 Amplify의 AWS CodeBuild 컨테이너는 사용자를 대신하여이 캐시를 압축하고 Amazon S3에 업로드하여 후속 빌드 성능을 개선합니다. 이로 인해 메모리 부족 오류가 발생하면 빌드가 실패할 수 있습니다.

빌드 단계에서 앱이 메모리 제한을 초과하지 않도록 하려면 다음 작업을 수행합니다. 먼저 빌드 설정의 cache.paths 섹션에서 `.next/cache/**/*`를 삭제합니다. 그런 다음, 빌드 설정 파일에서 `NODE_OPTIONS` 환경 변수를 제거합니다. 대신 Amplify 콘솔에서 `NODE_OPTIONS` 환경 변수를 설정하여 노드 최대 메모리 제한을 정의합니다. Amplify 콘솔을 사용하여 환경 변수를 설정하는 방법에 대한 자세한 내용은 환경 변수 설정을 참조하십시오.

이러한 변경을 수행한 후 빌드를 다시 시도합니다. 빌드가 성공하면 빌드 설정 파일의 cache.paths 섹션에 `.next/cache/**/*`를 다시 추가합니다.

빌드 성능을 개선하기 위한 Next.js 캐시 구성에 대한 자세한 내용은 Next.js 웹사이트의 AWS CodeBuild를 참조하십시오.

애플리케이션에 SSR 브랜치와 SSG 브랜치가 모두 있음

SSR 브랜치와 SSG 브랜치가 모두 있는 앱은 배포할 수 없습니다. SSR 브랜치와 SSG 브랜치를 모두 배포해야 하는 경우, SSR 브랜치만 사용하는 하나의 앱과 SSG 브랜치만 사용하는 다른 앱을 배포해야 합니다.

애플리케이션이 예약된 경로의 폴더에 정적 파일을 저장 중임

Next.js는 프로젝트의 루트 디렉터리에 이름이 `public`으로 지정되어 저장된 폴더의 정적 파일을 제공할 수 있습니다. Amplify를 사용하여 Next.js 앱을 배포하고 호스팅하는 경우 프로젝트에 `public/static` 경로가 있는 폴더를 포함할 수 없습니다. Amplify는 앱을 배포할 때 사용할 `public/static` 경로를 예약합니다. 앱에 이 경로가 포함된 경우, Amplify로 배포하기 전에 `static` 폴더 이름을 변경해야 합니다.

애플리케이션이 CloudFront 한도에 도달함

CloudFront 서비스 할당량은 Lambda@Edge 함수가 연결된 25개의 배포로 AWS 계정을 제한합니다. 이 할당량을 초과하는 경우, 사용하지 않는 CloudFront 배포를 계정에서 삭제하거나 할당량 증가를 요청할 수 있습니다. 자세한 내용은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하십시오.

Lambda@Edge 함수가 미국 동부(버지니아 북부) 리전에서 생성됨

Next.js 앱 배포 시 Amplify는 CloudFront를 통해 전달되는 콘텐츠를 사용자 지정할 수 있도록 Lambda@Edge 함수를 생성합니다. Lambda@Edge 함수는 앱이 배포된 리전이 아닌 미국 동부(버지니아 북부) 리전에서 생성됩니다. 이는 Lambda@Edge의 제한 사항입니다. Lambda@Edge 함수에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 엣지 함수 제한을 참조하십시오.

Next.js 애플리케이션이 지원되지 않는 기능을 사용함

Amplify로 배포된 앱은 버전 11까지의 Next.js 메이저 버전을 지원합니다. Amplify에서 지원되거나 지원되지 않는 Next.js 기능의 상세 목록은 supported features 섹션을 참조하십시오.

새 Next.js 앱 배포 시 Amplify는 지원되는 가장 최신 버전의 Next.js를 기본적으로 사용합니다. 이전 버전의 Next.js 버전으로 Amplify에 배포한 기존 Next.js 앱이 있는 경우, 이 앱을 Amplify 호스팅 컴퓨팅 SSR 공급자로 마이그레이션할 수 있습니다. 지침은 Next.js 11 SSR 앱을 Amplify Hosting 컴퓨팅으로 마이그레이션 섹션을 참조하세요.

Next.js 애플리케이션의 이미지가 로드되지 않음

`next/image` 구성 요소를 사용하여 Next.js 앱에 이미지를 추가할 때, 이미지 크기는 1MB를 초과할 수 없습니다. Amplify에 앱을 배포할 때 이미지가 1MB보다 큰 경우 503 오류를 반환합니다. 이는 헤더 및 본문을 포함하여 Lambda 함수가 생성하는 응답의 크기를 1MB로 제한하는 Lambda@Edge 제한 때문입니다.

1MB 제한은 PDF, 문서 파일 등 앱의 다른 아티팩트에 적용됩니다.

지원되지 않는 리전

Amplify는 Amplify를 사용할 수 있는 모든 AWS 리전에서 Classic(Next.js 11만 해당) SSR 앱 배포를 지원하지 않습니다. Classic(Next.js 11만 해당) SSR은 유럽(밀라노) eu-south-1, 중동(바레인) me-south-1, 아시아 태평양(홍콩) ap-east-1 리전에서 지원되지 않습니다.

SSR 앱 요금

SSR 앱을 배포하는 경우 Amplify Hosting 컴퓨팅은 SSR 앱을 배포하는 데 필요한 리소스를 관리합니다. Amplify Hosting 컴퓨팅 요금에 대한 자세한 내용은 AWS Amplify 요금을 참조하십시오.

SSR 배포 문제 해결

Amplify Hosting 컴퓨팅을 사용하여 SSR 앱을 배포할 때 예상치 못한 문제가 발생하는 경우, Amplify 문제 해결 장의 서버 측 렌더링 애플리케이션 문제 해결 섹션을 참조하세요.

고급: 오픈 소스 어댑터

프레임워크 작성자는 파일 시스템 기반 배포 사양을 참조하여 특정 프레임워크용으로 사용자 지정한 오픈 소스 빌드 어댑터를 개발할 수 있습니다. 이러한 어댑터에서는 앱의 빌드 출력이 Amplify Hosting의 예상 디렉터리 구조에 부합하는 배포 번들로 변환됩니다. 이 배포 번들에는 라우팅 규칙과 같은 런타임 구성을 비롯해 앱을 호스팅하는 데 필요한 모든 파일과 자산이 포함됩니다.

프레임워크를 사용하지 않는다면 자체 솔루션을 개발하여 Amplify의 예상 빌드 출력을 생성할 수 있습니다.

주제

- Amplify Hosting 배포 사양을 참조하여 빌드 출력 구성
- 배포 매니페스트를 사용하여 Express 서버 배포
- 프레임워크 작성자를 위한 이미지 최적화 통합
- 모든 SSR 프레임워크에 오픈 소스 어댑터 사용

Amplify Hosting 배포 사양을 참조하여 빌드 출력 구성

Amplify Hosting 배포 사양은 Amplify Hosting으로 배포를 촉진하도록 디렉터리 구조를 정의하는 파일 시스템 기반 사양입니다. 프레임워크에서는 Amplify Hosting의 서비스 기본 요소가 프레임워크에서 활용되도록 이러한 예상 디렉터리 구조를 빌드 명령의 출력으로 생성할 수 있습니다. Amplify Hosting은 배포 번들의 구조를 파악하고 적절히 배포합니다.

배포 사양을 사용하는 방법을 설명하는 동영상 데모는 Amazon Web Services YouTube 채널에서 How to host any website using AWS Amplify를 시청하세요.

다음은 Amplify에서 예상하는 배포 번들 폴더 구조의 예시입니다. 상위 수준에는 이름이 `static`인 폴더, 이름이 `compute`인 폴더 및 이름이 `deploy-manifest.json`인 배포 매니페스트 파일이 있습니다.

```
.amplify-hosting/
```

```
### compute/
#   ### default/
#       ### chunks/
#       #   ### app/
#       #       ### _nuxt/
#       #       #   ### index-xxx.mjs
#       #       #   ### index-styles.xxx.js
#       #       ### server.mjs
#       ### node_modules/
#       ### server.js
### static/
#   ### css/
#   #   ### nuxt-google-fonts.css
#   ### fonts/
#   #   ### font.woff2
#   ### _nuxt/
#   #   ### builds/
#   #   #   ### latest.json
#   #   ### entry.xxx.js
#   ### favicon.ico
#   ### robots.txt
### deploy-manifest.json
```

Amplify SSR 기본 요소 지원

Amplify Hosting 배포 사양에는 다음과 같은 기본 요소에 긴밀하게 매핑되는 계약이 정의됩니다.

정적 자산

정적 파일을 호스팅하는 기능을 프레임워크에 제공합니다.

컴퓨팅

포트 3000에서 Node.js HTTP 서버를 실행하는 기능을 프레임워크에 제공합니다.

이미지 최적화

런타임 시 이미지를 최적화하는 서비스를 프레임워크에 제공합니다.

라우팅 규칙

수신되는 요청 경로를 특정 대상에 매핑하는 메커니즘을 프레임워크에 제공합니다.

.amplify-hosting/static 디렉터리

애플리케이션 URL에서 제공하기로 되어 있으며 공개적으로 액세스할 수 있는 모든 정적 파일을 `.amplify-hosting/static` 디렉터리에 배치해야 합니다. 이 디렉터리 내부의 파일은 정적 자산 기본 요소를 통해 제공됩니다.

정적 파일은 콘텐츠, 파일 이름 또는 확장명을 변경하지 않고 애플리케이션 URL의 루트(/)에서 액세스할 수 있습니다. 또한 하위 디렉터리는 URL 구조에 보존되며 파일 이름 앞에 표시됩니다. 예를 들면 `.amplify-hosting/static/favicon.ico`는 `https://myAppId.amplify-hostingapp.com/favicon.ico`에서 제공되고 `.amplify-hosting/static/_nuxt/main.js`는 `https://myAppId.amplify-hostingapp.com/_nuxt/main.js`에서 제공됩니다.

프레임워크에서 애플리케이션의 기본 경로를 수정하는 기능이 지원되면 `.amplify-hosting/static` 디렉터리 내부의 정적 자산 앞에 기본 경로를 추가해야 합니다. 예를 들어 기본 경로가 `/folder1/folder2`인 경우에는 `main.css`라는 정적 자산의 빌드 출력이 `.amplify-hosting/static/folder1/folder2/main.css`가 됩니다.

.amplify-hosting/compute 디렉터리

단일 컴퓨팅 리소스는 `.amplify-hosting/compute` 디렉터리 내에 포함된 `default`라는 단일 하위 디렉터리로 표시됩니다. 경로는 `.amplify-hosting/compute/default`입니다. 이 컴퓨팅 리소스는 Amplify Hosting의 컴퓨팅 기본 요소에 매핑됩니다.

`default` 하위 디렉터리의 콘텐츠는 다음 규칙을 준수해야 합니다.

- 컴퓨팅 리소스의 진입점 역할을 하려면 파일이 `default` 하위 디렉터리의 루트에 있어야 합니다.
- 진입점 파일은 Node.js 모듈이어야 하며 포트 3000에서 수신 대기하는 HTTP 서버가 시작되어야 합니다.
- 다른 파일을 `default` 하위 디렉터리에 배치하고 진입점 파일의 코드에서 해당 파일을 참조할 수 있습니다.
- 하위 디렉터리의 콘텐츠는 독립적이어야 합니다. 진입점 모듈의 코드에서는 하위 디렉터리 외부의 모듈을 참조할 수 없습니다. 참고로, 프레임워크에서는 원하는 방식으로 HTTP 서버를 번들링할 수 있습니다. 하위 디렉터리 내에서 항목 파일의 이름이 `server.js` is인 `node server.js` 명령으로 컴퓨팅 프로세스를 시작할 수 있으면 Amplify에서는 디렉터리 구조가 배포 사양을 준수하는 것으로 간주합니다.

Amplify Hosting에서는 `default` 하위 디렉터리 내부의 모든 파일을 번들링하여 프로비저닝한 컴퓨팅 리소스에 배포합니다. 각 컴퓨팅 리소스에는 512MB의 임시 스토리지가 할당됩니다. 이 스토리지

는 실행 인스턴스 간에 공유되지 않지만, 동일한 실행 인스턴스 내의 후속 간접 호출 간에는 공유됩니다. 실행 인스턴스의 최대 실행 시간은 15분으로 제한되며, 실행 인스턴스 내 쓰기 가능한 경로는 `/tmp` 디렉터리뿐입니다. 각 컴퓨팅 리소스 번들의 압축된 크기는 220MB를 초과할 수 없습니다. 예를 들어 `.amplify/compute/default` 하위 디렉터리는 압축 시 220MB를 초과할 수 없습니다.

.amplify-hosting/deploy-manifest.json 파일

`deploy-manifest.json` 파일을 사용하여 배포의 구성 세부 정보와 메타데이터를 저장합니다. `deploy-manifest.json` 파일에는 최소한 `version` 속성, catch-all 라우팅이 지정된 `routes` 속성, 프레임워크 메타데이터가 지정된 `framework` 속성이 포함되어야 합니다.

다음 객체 정의에서는 배포 매니페스트의 구성을 보여줍니다.

```
type DeployManifest = {
  version: 1;
  routes: Route[];
  computeResources?: ComputeResource[];
  imageSettings?: ImageSettings;
  framework: FrameworkMetadata;
};
```

다음 주제에서는 배포 매니페스트의 각 속성에 대한 세부 정보 및 사용법을 설명합니다.

version 속성 사용

`version` 속성에서는 구현 중인 배포 사양의 버전이 정의됩니다. 현재 Amplify Hosting 배포 사양의 버전은 버전 1뿐입니다. 다음 JSON 예시에서는 `version` 속성의 사용법을 보여줍니다.

```
"version": 1
```

routes 속성 사용

`routes` 속성을 통해 프레임워크에서 Amplify Hosting 라우팅 규칙 기본 요소를 활용할 수 있습니다. 라우팅 규칙은 수신되는 요청 경로를 배포 번들의 특정 대상으로 라우팅하는 메커니즘을 제공합니다. 라우팅 규칙은 수신되는 요청의 목적지만 지정하며, 다시 쓰기 및 리디렉션 규칙에 따라 요청이 변환된 후에 라우팅 규칙이 적용됩니다. Amplify Hosting에서 다시 쓰기와 리디렉션을 처리하는 방식에 대한 자세한 내용은 Amplify 애플리케이션에 대한 리디렉션 및 다시 쓰기 설정 섹션을 참조하세요.

라우팅 규칙은 요청을 다시 쓰거나 변환하지 않습니다. 수신되는 요청이 라우팅의 경로 패턴과 일치하면 요청이 그대로 대상에 라우팅됩니다.

`routes` 배열에 지정된 라우팅 규칙에서는 다음과 같은 규칙을 준수해야 합니다.

- catch-all 라우팅이 지정되어야 합니다. catch-all 라우팅에는 수신되는 모든 요청과 일치하는 `/*` 패턴이 있습니다.
- `routes` 배열에는 최대 25개 항목이 포함될 수 있습니다.
- `Static` 경로 또는 `Compute` 경로를 지정해야 합니다.
- `Static` 경로를 지정하는 경우 `.amplify-hosting/static` 디렉터리가 있어야 합니다.
- `Compute` 경로를 지정하는 경우 `.amplify-hosting/compute` 디렉터리가 있어야 합니다.
- `ImageOptimization` 경로를 지정하는 경우 `Compute` 경로도 지정해야 합니다. 완전히 정적인 애플리케이션에는 아직 이미지 최적화가 지원되지 않기 때문에 이 작업이 필요합니다.

다음 객체 정의에서는 `Route` 객체의 구성을 보여줍니다.

```
type Route = {
  path: string;
  target: Target;
  fallback?: Target;
}
```

다음 테이블에는 `Route` 객체의 속성이 설명되어 있습니다.

키	유형	필수	설명
경로	String	예	수신되는 요청 경로와 일치하는 패턴을 정의합니다(쿼리 문자열 제외). 최대 경로 길이는 255자입니다. 경로는 슬래시(/)로 시작해야 합니다. 경로에는 [A-Z], [a-z], [0-9], [_-.*$/~"'@:+] 문자가 포함될 수 있습니다.

키	유형	필수	설명
			패턴 일치에는 다음과 같은 와일드카드 문자만 지원됩니다. • *(0개 이상의 문자와 일치) • 이 /* 패턴은 catch-all 패턴이라고 하며 수신되는 모든 요청과 일치합니다.
대상	대상	예	일치하는 요청을 라우팅할 대상이 정의되는 객체입니다. Compute 경로를 지정하면 해당하는 ComputeResource 가 있어야 합니다. ImageOptimization 경로를 지정하면 imageSettings 도 지정해야 합니다.

키	유형	필수	설명
fallback	대상	아니요	원래 대상에서 404 오류가 반환되는 경우 대체할 대상이 정의되는 객체입니다. 지정한 경로의 `target` 종류와 `fallback` 종류는 같을 수 없습니다. 예를 들면 `Static`에서 `Static`으로의 대체는 허용되지 않습니다. 본문이 없는 GET 요청에만 대체가 지원됩니다. 요청에 본문이 있으면 대체 중에 본문이 삭제됩니다.

다음 객체 정의에서는 `Target` 객체의 구성을 보여줍니다.

```
type Target = {
  kind: TargetKind;
  src?: string;
  cacheControl?: string;
}
```

다음 테이블에는 `Target` 객체의 속성이 설명되어 있습니다.

키	유형	필수	설명
kind	Targetkind	예	대상 유형이 정의되는 `enum`입니다. 유효한 값은 `Static`, `Compute`,

키	유형	필수	설명
			`ImageOptimization` 입니다.
src	String	`Compute`의 경우 예 기타 기본 요소의 경우 아니요	기본 요소의 실행 코드가 포함되어 있는 배포 번들의 하위 디렉터리 이름이 지정되는 문자열입니다. 컴퓨팅 기본 요소의 경우에만 유효하고 필요합니다. 값은 배포 번들에 있는 컴퓨팅 리소스 중 하나를 가리켜야 합니다. 현재 이 필드에서 지원되는 유일한 값은 `default`입니다.

키	유형	필수	설명
cacheControl	String	No	응답에 적용할 Cache-Control 헤더의 값을 지정하는 문자열입니다. Static 및 ImageOptimization 기본 요소에만 유효합니다. 지정한 값은 사용자 지정 헤더에 의해 재정의됩니다. Amplify Hosting 사용자 지정 헤더에 대한 자세한 내용은 Amplify 앱에 대한 사용자 지정 HTTP 헤더 설정 섹션을 참조하세요. **Note** 이 Cache-Control 헤더는 상태 코드가 200(OK)으로 설정된 성공 응답에만 적용됩니다.

다음 객체 정의에서는 `TargetKind` 열거의 사용법을 보여줍니다.

```
enum TargetKind {
  Static = "Static",
  Compute = "Compute",
  ImageOptimization = "ImageOptimization"
```

```
}
```

다음 목록을 통해 `TargetKind` 열거형의 유효한 값이 지정됩니다.

정적

요청을 정적 자산 기본 요소로 라우팅합니다.

컴퓨팅

요청을 컴퓨팅 기본 요소로 라우팅합니다.

ImageOptimization

요청을 이미지 최적화 기본 요소로 라우팅합니다.

다음 JSON 예시에서는 여러 라우팅 규칙이 지정된 `routes` 속성의 사용법을 보여줍니다.

```
"routes": [
    {
      "path": "/_nuxt/image",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=3600, immutable"
      }
    },
    {
      "path": "/_nuxt/builds/meta/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/_nuxt/builds/*",
      "target": {
        "cacheControl": "public, max-age=1, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/_nuxt/*",
      "target": {
```

```
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/*.*",
      "target": {
        "kind": "Static"
      },
      "fallback": {
        "kind": "Compute",
        "src": "default"
      }
    },
    {
      "path": "/*",
      "target": {
        "kind": "Compute",
        "src": "default"
      }
    }
  ]
```

배포 매니페스트의 라우팅 규칙 지정에 대한 자세한 내용은 라우팅 규칙 구성 모범 사례 섹션을 참조하세요.

computeResources 속성 사용

프로비저닝한 컴퓨팅 리소스에 대한 메타데이터가 이 `computeResources` 속성을 통해 프레임워크에서 제공될 수 있습니다. 모든 컴퓨팅 리소스는 해당하는 경로와 연결되어 있어야 합니다.

다음 객체 정의에서는 `ComputeResource` 객체의 사용법을 보여줍니다.

```
type ComputeResource = {
  name: string;
  runtime: ComputeRuntime;
  entrypoint: string;
};

type ComputeRuntime = 'nodejs16.x' | 'nodejs18.x' | 'nodejs20.x';
```

다음 테이블에는 `ComputeResource` 객체의 속성이 설명되어 있습니다.

키	유형	필수	설명
name	String	예	컴퓨팅 리소스의 이름을 지정합니다. 이름은 `.amplify-hosting/compute` directory 내부의 하위 디렉터리 이름과 일치해야 합니다. 배포 사양 버전 1의 경우 유효한 값은 `default`뿐입니다.
런타임	ComputeRuntime	예	프로비저닝한 컴퓨팅 리소스의 런타임을 정의합니다. 유효한 값은 `nodejs16.x` , `nodejs18.x` , `nodejs20.x` 입니다.
entrypoint	String	예	지정한 컴퓨팅 리소스에 대해 코드가 실행될 시작 파일의 이름을 지정합니다. 파일은 컴퓨팅 리소스를 나타내는 하위 디렉터리 내부에 있어야 합니다.

다음과 같은 디렉터리 구조가 있는 경우입니다.

```
.amplify-hosting
|---compute
|   |---default
```

```
|       |---index.js
```

computeResource 속성의 JSON은 다음과 같습니다.

```
"computeResources": [
    {
      "name": "default",
      "runtime": "nodejs16.x",
      "entrypoint": "index.js",
    }
  ]
```

imageSettings 속성 사용

이 imageSettings 속성을 통해 런타임 시 이미지의 온디맨드 최적화가 제공되는 이미지 최적화 기본 요소의 동작을 프레임워크에서 사용자 지정할 수 있습니다.

다음 객체 정의에서는 ImageSettings 객체의 사용법을 보여줍니다.

```
type ImageSettings = {
  sizes: number[];
  domains: string[];
  remotePatterns: RemotePattern[];
  formats: ImageFormat[];
  minumumCacheTTL: number;
  dangerouslyAllowSVG: boolean;
};

type ImageFormat = 'image/avif' | 'image/webp' | 'image/png' | 'image/jpeg';
```

다음 테이블에는 ImageSettings 객체의 속성이 설명되어 있습니다.

키	유형	필수	설명
크기	Number[]	예	지원되는 이미지 너비의 배열입니다.
domains	String[]	예	이미지 최적화를 사용하도록 허용된 외부 도메인의 배열입니다. 배

키	유형	필수	설명
			포 도메인에서만 이미지 최적화가 사용되도록 하려면 배열을 비워 둡니다.
remotePatterns	RemotePattern[]	예	이미지 최적화를 사용하도록 허용된 외부 패턴의 배열입니다. 도메인과 비슷하지만, 정규 표현식(regex)이 제공되어 추가로 제어할 수 있습니다.
formats	ImageFormat[]	예	허용된 출력 이미지 형식의 배열입니다.
minimumCacheTTL	숫자	예	최적화된 이미지의 캐시 기간(초)입니다.
dangerouslyAllowSVG	불	예	SVG 입력 이미지 URL을 허용합니다. 보안을 위해 기본적으로 비활성화되어 있습니다.

다음 객체 정의에서는 `RemotePattern` 객체의 사용법을 보여줍니다.

```
type RemotePattern = {
  protocol?: 'https';
  hostname: string;
  port?: string;
  pathname?: string;
}
```

다음 테이블에는 `RemotePattern` 객체의 속성이 설명되어 있습니다.

키	유형	필수	설명
프로토콜	String	No	허용된 원격 패턴의 프로토콜입니다. 유일한 유효 값은 `https`입니다.
hostname	String	예	허용된 원격 패턴의 호스트 이름입니다. 리터럴 또는 와일드카드를 지정할 수 있습니다. 1개(`*`)는 하나의 하위 도메인과 일치합니다. 2개(`**`)는 모든 하위 도메인의 수와 일치합니다. Amplify에서는 `**`만 지정하는 포괄적인 와일드카드가 허용되지 않습니다.
포트	String	No	허용된 원격 패턴의 포트입니다.
pathname	String	No	허용된 원격 패턴의 경로 이름입니다.

다음 예시에서는 `imageSettings` 속성을 보여줍니다.

```
"imageSettings": {
    "sizes": [
      100,
      200
    ],
    "domains": [
      "example.com"
    ],
    "remotePatterns": [
```

```
      {
        "protocol": "https",
        "hostname": "example.com",
        "port": "",
        "pathname": "/**",
      }
    ],
    "formats": [
      "image/webp"
    ],
    "minumumCacheTTL": 60,
    "dangerouslyAllowSVG": false
  }
```

framework 속성 사용

`framework` 속성을 사용하여 프레임워크 메타데이터를 지정합니다.

다음 객체 정의에서는 `FrameworkMetadata` 객체의 구성을 보여줍니다.

```
type FrameworkMetadata = {
  name: string;
  version: string;
}
```

다음 테이블에는 `FrameworkMetadata` 객체의 속성이 설명되어 있습니다.

키	유형	필수	설명
name	String	예	프레임워크의 이름입니다.
version	String	예	프레임워크의 버전입니다. 유효한 의미 체계 버전 관리(semver) 문자열이어야 합니다.

라우팅 규칙 구성 모범 사례

라우팅 규칙에서는 수신되는 요청 경로를 배포 번들의 특정 대상으로 라우팅하는 메커니즘이 제공됩니다. 배포 번들에서 프레임워크 작성자는 다음과 같은 대상 중 하나에 배포되는 파일을 빌드 출력에 내보낼 수 있습니다.

- 정적 자산 기본 요소 – 파일이 `.amplify-hosting/static` 디렉터리에 포함되어 있습니다.
- 컴퓨팅 기본 요소 – 파일이 `.amplify-hosting/compute/default` 디렉터리에 포함되어 있습니다.

프레임워크 작성자는 라우팅 규칙 배열도 배포 매니페스트 파일로 제공합니다. 배열의 각 규칙은 일치 항목이 있을 때까지 수신되는 요청과 순차적인 순회 순서로 대조됩니다. 일치하는 규칙이 있으면 일치하는 규칙에 지정된 대상에 요청이 라우팅됩니다. 규칙마다 대체 대상을 지정할 수도 있습니다. 원래 대상에서 404 오류를 반환하면 요청이 대체 대상으로 라우팅됩니다.

배포 사양에서는 순회 순서의 마지막 규칙이 catch-all 규칙이 되어야 합니다. catch-all 규칙은 `/*` 경로와 함께 지정됩니다. 수신되는 요청이 라우팅 규칙 배열의 이전 경로와 전혀 일치하지 않으면 요청이 catch-all 규칙 대상으로 라우팅됩니다.

SSR 프레임워크(예: Nuxt.js)의 경우 catch-all 규칙 대상이 컴퓨팅 기본 요소여야 합니다. SSR 애플리케이션에는 빌드 시 예측할 수 없는 경로로 렌더링된 서버 측 페이지가 있기 때문입니다. 예를 들어 `/blog/[slug]`에 Nuxt.js 애플리케이션의 페이지가 있으면 `[slug]`가 동적 경로 파라미터입니다. catch-all 규칙 대상은 요청을 이러한 페이지에 라우팅하는 유일한 방법입니다.

이와 달리 특정 경로 패턴을 사용하여 빌드 시 알려진 경로를 대상으로 지정할 수 있습니다. 예를 들면 `/_nuxt`에서는 정적 자산이 Nuxt.js 경로에서 제공됩니다. 따라서 요청을 정적 자산 기본 요소로 라우팅하는 특정 라우팅 규칙에 따라 `/_nuxt/*` 경로를 대상으로 지정할 수 있습니다.

퍼블릭 폴더 라우팅

대다수 SSR 프레임워크는 `public` 폴더에 있는 변경 가능한 정적 자산을 사용하는 기능이 제공됩니다. `favicon.ico` 및 `robots.txt`와 같은 파일은 일반적으로 `public` 내부에 유지되며 애플리케이션의 루트 URL에서 제공됩니다. 예를 들면 `favicon.ico` 파일은 `https://example.com/favicon.ico`에서 제공됩니다. 참고로, 이러한 파일에는 예측 가능한 경로 패턴이 없습니다. 거의 전적으로 파일 이름에 따라 결정됩니다. `public` 폴더 내부의 파일을 대상으로 지정하는 유일한 방법은 catch-all 라우팅을 사용하는 것입니다. 그러나 catch-all 라우팅 대상은 컴퓨팅 기본 요소여야 합니다.

`public` 폴더 관리에는 다음과 같은 접근 방식 중 하나를 따르는 것이 좋습니다.

1. 경로 패턴을 사용하여 파일 확장명이 있는 요청 경로를 대상으로 지정합니다. 예를 들면 `/*.*`를 사용하여 파일 확장명이 있는 모든 요청 경로를 대상으로 지정할 수 있습니다.

   참고로, 이 접근 방식은 신뢰할 수 없습니다. 예를 들면 public 폴더에 속한 파일에 파일 확장명이 없다면 해당 파일은 이 규칙에 따라 대상으로 지정되지 않습니다. 이 접근 방식에서 유의할 다른 문제는 이름에 마침표가 있는 페이지가 애플리케이션에 있을 수 있다는 점입니다. 예를 들면 /blog/2021/01/01/hello.world의 페이지가 `/*.*` 규칙에 따라 대상으로 지정됩니다. 페이지는 정적 자산이 아니므로 이는 바람직하지 않습니다. 그러나 정적 기본 요소에서 404 오류가 발생하는 경우 요청이 컴퓨팅 기본 요소로 대체되도록 이 규칙에 대체 대상을 추가할 수 있습니다.

   ```
   {
       "path": "/*.*",
       "target": {
           "kind": "Static"
       },
       "fallback": {
           "kind": "Compute",
           "src": "default"
       }
   }
   ```

2. 빌드 시 public 폴더의 파일을 식별하고 각 파일에 대한 라우팅 규칙을 내보냅니다. 배포 사양에 따라 규칙이 25개로 제한되므로 이 접근 방식은 확장할 수 없습니다.

   ```
   {
       "path": "/favicon.ico",
       "target": {
           "kind": "Static"
       }
   },
   {
       "path": "/robots.txt",
       "target": {
           "kind": "Static"
       }
   }
   ```

3. 프레임워크 사용자는 변경 가능한 모든 정적 자산을 public 폴더의 하위 폴더에 저장하는 것이 좋습니다.

다음 예시에서는 사용자가 변경 가능한 모든 정적 자산을 public/assets 폴더에 저장할 수 있습니다. 그런 다음 경로 패턴이 /assets/*인 라우팅 규칙을 사용하여 public/assets 폴더의 모든 변경 가능한 정적 자산을 대상으로 지정할 수 있습니다.

```
{
    "path": "/assets/*",
    "target": {
        "kind": "Static"
    }
}
```

4. catch-all 라우팅에 대한 정적 대체를 지정합니다. 이 접근 방식에는 단점이 있으며 다음 catch-all 대체 라우팅 섹션에 자세히 설명되어 있습니다.

catch-all 대체 라우팅

컴퓨팅 기본 요소 대상에 대해 catch-all 라우팅이 지정된 SSR 프레임워크(예: Nuxt.js)의 경우 프레임워크 작성자가 public 폴더 라우팅 문제를 해결하려면 catch-all 라우팅에 대한 정적 대체를 지정하는 것이 좋습니다. 그러나 이 라우팅 규칙 유형은 서버 측에서 렌더링한 404 페이지를 중단시킵니다. 예를 들어 최종 사용자가 존재하지 않는 페이지를 방문하면 애플리케이션에서는 상태 코드가 404인 404 페이지를 렌더링합니다. 그러나 catch-all 라우팅에 정적 대체가 있으면 404 페이지가 렌더링되지 않습니다. 대신에 요청은 정적 기본 요소로 대체되고 여전히 404 상태 코드로 끝나지만, 404 페이지는 렌더링되지 않습니다.

```
{
    "path": "/*",
    "target": {
        "kind": "Compute",
        "src": "default"
    },
    "fallback": {
        "kind": "Static"
    }
}
```

기본 경로 라우팅

애플리케이션의 기본 경로를 수정하는 기능이 제공되는 프레임워크에서는 .amplify-hosting/static 디렉터리 내부의 정적 자산 앞에 기본 경로를 추가할 것으로 예상됩니다. 예를 들어 기본 경로

가 /folder1/folder2인 경우에는 main.css라는 정적 자산의 빌드 출력이 .amplify-hosting/static/folder1/folder2/main.css가 됩니다.

따라서 기본 경로가 반영되도록 라우팅 규칙도 업데이트해야 합니다. 예를 들어 기본 경로가 /folder1/folder2인 경우에는 public 폴더의 정적 자산에 대한 라우팅 규칙은 다음과 같습니다.

```
{
    "path": "/folder1/folder2/*.*",
    "target": {
        "kind": "Static"
    }
}
```

마찬가지로 서버 측 경로도 앞에 기본 경로를 추가해야 합니다. 예를 들어 기본 경로가 /folder1/folder2인 경우에는 /api 경로에 대한 라우팅 규칙은 다음과 같습니다.

```
{
    "path": "/folder1/folder2/api/*",
    "target": {
        "kind": "Compute",
        "src": "default"
    }
}
```

그러나 catch-all 라우팅 앞에는 기본 경로를 추가하면 안 됩니다. 예를 들어 기본 경로가 /folder1/folder2인 경우에는 catch-all 라우팅이 다음과 같이 유지됩니다.

```
{
    "path": "/*",
    "target": {
        "kind": "Compute",
        "src": "default"
    }
}
```

Nuxt.js 경로 예시

다음은 라우팅 규칙을 지정하는 방법을 보여주는 Nuxt 애플리케이션의 예시 deploy-manifest.json 파일입니다.

```
{
```

```
  "version": 1,
  "routes": [
    {
      "path": "/_nuxt/image",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=3600, immutable"
      }
    },
    {
      "path": "/_nuxt/builds/meta/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/_nuxt/builds/*",
      "target": {
        "cacheControl": "public, max-age=1, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/_nuxt/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/*.*",
      "target": {
        "kind": "Static"
      },
      "fallback": {
        "kind": "Compute",
        "src": "default"
      }
    },
    {
      "path": "/*",
      "target": {
        "kind": "Compute",
```

```
        "src": "default"
      }
    }
  ],
  "computeResources": [
    {
      "name": "default",
      "entrypoint": "server.js",
      "runtime": "nodejs18.x"
    }
  ],
  "framework": {
    "name": "nuxt",
    "version": "3.8.1"
  }
}
```

다음은 기본 경로가 포함된 라우팅 규칙을 지정하는 방법을 보여주는 Nuxt의 예시 `deploy-manifest.json` 파일입니다.

```
{
  "version": 1,
  "routes": [
    {
      "path": "/base-path/_nuxt/image",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=3600, immutable"
      }
    },
    {
      "path": "/base-path/_nuxt/builds/meta/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/base-path/_nuxt/builds/*",
      "target": {
        "cacheControl": "public, max-age=1, immutable",
        "kind": "Static"
      }
```

```
    },
    {
      "path": "/base-path/_nuxt/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/base-path/*.*",
      "target": {
        "kind": "Static"
      },
      "fallback": {
        "kind": "Compute",
        "src": "default"
      }
    },
    {
      "path": "/*",
      "target": {
        "kind": "Compute",
        "src": "default"
      }
    }
  ],
  "computeResources": [
    {
      "name": "default",
      "entrypoint": "server.js",
      "runtime": "nodejs18.x"
    }
  ],
  "framework": {
    "name": "nuxt",
    "version": "3.8.1"
  }
}
```

`routes` 속성에 대한 자세한 내용은 routes 속성 사용 섹션을 참조하세요.

배포 매니페스트를 사용하여 Express 서버 배포

이 예시에서는 Amplify Hosting 배포 사양을 참조하여 기본 Express 서버를 배포하는 방법을 설명합니다. 제공된 배포 매니페스트를 활용하여 라우팅, 컴퓨팅 리소스 및 기타 구성을 지정할 수 있습니다.

Amplify Hosting에 배포하기 전에 로컬로 Express 서버 설정

1. 프로젝트의 새 디렉터리를 만들고 Express와 Typescript를 설치합니다.

```
mkdir express-app
cd express-app

# The following command will prompt you for information about your project
npm init

# Install express, typescript and types
npm install express --save
npm install typescript ts-node @types/node @types/express --save-dev
```

2. 프로젝트의 루트에 다음과 같은 콘텐츠가 포함된 `tsconfig.json` 파일을 추가합니다.

```
{
  "compilerOptions": {
    "target": "es6",
    "module": "commonjs",
    "outDir": "./dist",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true
  },
  "include": ["src/**/*.ts"],
  "exclude": ["node_modules"]
}
```

3. 프로젝트 루트에 `src`라는 디렉터리를 만듭니다.
4. `src` 디렉터리에 `index.ts` 파일을 만듭니다. 이 파일이 애플리케이션에서 Express 서버가 시작되는 진입점이 됩니다. 포트 3000에서 수신 대기하도록 서버를 구성해야 합니다.

```
// src/index.ts
import express from 'express';
```

```
const app: express.Application = express();
const port = 3000;

app.use(express.text());

app.listen(port, () => {
  console.log(`server is listening on ${port}`);
});

// Homepage
app.get('/', (req: express.Request, res: express.Response) => {
  res.status(200).send("Hello World!");
});

// GET
app.get('/get', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-get-header", "get-header-value").send("get-response-
from-compute");
});

//POST
app.post('/post', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-post-header", "post-header-
value").send(req.body.toString());
});

//PUT
app.put('/put', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-put-header", "put-header-
value").send(req.body.toString());
});

//PATCH
app.patch('/patch', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-patch-header", "patch-header-
value").send(req.body.toString());
});

// Delete
app.delete('/delete', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-delete-header", "delete-header-value").send();
});
```

5. 다음과 같은 스크립트를 package.json 파일에 추가합니다.

```
"scripts": {
  "start": "ts-node src/index.ts",
  "build": "tsc",
  "serve": "node dist/index.js"
}
```

6. 프로젝트의 루트에 public이라는 디렉터리를 만듭니다. 그런 다음에 다음과 같은 콘텐츠가 포함된 hello-world.txt라는 파일을 만듭니다.

```
Hello world!
```

7. 프로젝트 루트에 다음과 같은 콘텐츠가 포함된 .gitignore 파일을 추가합니다.

```
.amplify-hosting
dist
node_modules
```

Amplify 배포 매니페스트 설정

1. 프로젝트의 루트 디렉터리에 deploy-manifest.json이라는 파일을 만듭니다.
2. 다음 매니페스트를 복사하여 deploy-manifest.json 파일에 붙여넣습니다.

```
{
  "version": 1,
  "framework": { "name": "express", "version": "4.18.2" },
  "imageSettings": {
    "sizes": [
      100,
      200,
      1920
    ],
    "domains": [],
    "remotePatterns": [],
    "formats": [],
    "minimumCacheTTL": 60,
    "dangerouslyAllowSVG": false
  },
  "routes": [
    {
```

```
      "path": "/_amplify/image",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=3600, immutable"
      }
    },
    {
      "path": "/*.*",
      "target": {
        "kind": "Static",
        "cacheControl": "public, max-age=2"
      },
      "fallback": {
        "kind": "Compute",
        "src": "default"
      }
    },
    {
      "path": "/*",
      "target": {
        "kind": "Compute",
        "src": "default"
      }
    }
  ],
  "computeResources": [
    {
      "name": "default",
      "runtime": "nodejs18.x",
      "entrypoint": "index.js"
    }
  ]
}
```

매니페스트는 Amplify Hosting에서 애플리케이션 배포를 처리하는 방식을 설명합니다. 기본 설정은 다음과 같습니다.

- version – 사용 중인 배포 사양의 버전을 나타냅니다.
- framework – 이를 조정하여 Express 서버 설정을 지정합니다.
- imageSettings – 이미지 최적화를 처리하지 않는다면 이 섹션은 Express 서버에 대한 선택 사항입니다.

- routes – 앱의 적합한 부분으로 트래픽을 유도하는 데 매우 중요합니다. `"kind": "Compute"` 경로에서는 트래픽이 서버 로직으로 유도됩니다.
- computeResources – 이 섹션을 사용하여 Express 서버의 런타임과 진입점을 지정합니다.

다음으로, 빌드된 애플리케이션 아티팩트를 `.amplify-hosting` 배포 번들로 이동하는 빌드 후 스크립트를 설정합니다. 디렉터리 구조는 Amplify Hosting 배포 사양에 부합합니다.

빌드 후 스크립트 설정

1. 프로젝트 루트에 `bin`라는 디렉터리를 만듭니다.
2. `bin` 디렉터리에 `postbuild.sh`라는 파일을 만듭니다. 다음 내용을 `postbuild.sh` 파일에 추가합니다.

```
#!/bin/bash

rm -rf ./.amplify-hosting

mkdir -p ./.amplify-hosting/compute

cp -r ./dist ./.amplify-hosting/compute/default
cp -r ./node_modules ./.amplify-hosting/compute/default/node_modules

cp -r public ./.amplify-hosting/static

cp deploy-manifest.json ./.amplify-hosting/deploy-manifest.json
```

3. `package.json` 파일에 `postbuild` 스크립트를 추가합니다. 파일이 다음과 같아야 합니다.

```
"scripts": {
  "start": "ts-node src/index.ts",
  "build": "tsc",
  "serve": "node dist/index.js",
  "postbuild": "chmod +x bin/postbuild.sh && ./bin/postbuild.sh"
}
```

4. 다음 명령을 실행하여 애플리케이션을 빌드합니다.

```
npm run build
```

5. (선택 사항) Express의 경로를 조정합니다. Express 서버에 알맞게 배포 매니페스트의 경로를 수정할 수 있습니다. 예를 들어 `public` 디렉터리에 정적 자산이 없으면 Compute로 유도되는 catch-all 라우팅(`"path": "/*"`)만 필요할 수 있습니다. 이는 서버의 설정에 따라 다릅니다.

최종 디렉터리 구조는 다음과 같아야 합니다.

```
express-app/
### .amplify-hosting/
#   ### compute/
#   #   ### default/
#   #       ### node_modules/
#   #       ### index.js
#   ### static/
#   #   ### hello.txt
#   ### deploy-manifest.json
### bin/
#   ### .amplify-hosting/
#   #   ### compute/
#   #   #   ### default/
#   #   ### static/
#   ### postbuild.sh*
### dist/
#   ### index.js
### node_modules/
### public/
#   ### hello.txt
### src/
#   ### index.ts
### deploy-manifest.json
### package.json
### package-lock.json
### tsconfig.json
```

서버 배포

1. 코드를 Git 리포지토리로 푸시한 다음에 앱을 Amplify Hosting에 배포합니다.
2. 다음과 같이 `baseDirectory`에서 `.amplify-hosting`을 가리키도록 빌드 설정을 업데이트합니다. 빌드 중에 Amplify에서는 `.amplify-hosting` 디렉터리의 매니페스트 파일을 감지하고 Express 서버를 구성된 대로 배포합니다.

```
version: 1
```

```
frontend:
  phases:
    preBuild:
      commands:
        - nvm use 18
        - npm install
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .amplify-hosting
    files:
      - '**/*'
```

3. 배포가 성공했고 서버가 올바르게 실행되는지 확인하려면 Amplify Hosting에서 제공되는 기본 URL에서 앱을 방문합니다.

프레임워크 작성자를 위한 이미지 최적화 통합

프레임워크 작성자는 Amplify Hosting 배포 사양을 참조하여 Amplify의 이미지 최적화 기능을 통합할 수 있습니다. 이미지 최적화를 활성화하려면 배포 매니페스트에 이미지 최적화 서비스를 대상으로 지정하는 라우팅 규칙이 포함되어야 합니다. 다음 예시에서는 라우팅 규칙을 구성하는 방법을 보여줍니다.

```
// .amplify-hosting/deploy-manifest.json

{
  "routes": [
    {
      "path": "/images/*",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=31536000, immutable"
      }
    }
  ]
}
```

배포 사양을 참조하여 이미지 최적화 설정을 구성하는 방법에 대한 자세한 내용은 Amplify Hosting 배포 사양을 참조하여 빌드 출력 구성 섹션을 참조하세요.

이미지 최적화 API 이해

라우팅 규칙에 따라 정의된 경로에서 Amplify 앱의 도메인 URL을 통해 런타임에 이미지 최적화를 간접적으로 호출할 수 있습니다.

```
GET https://{appDomainName}/{path}?{queryParams}
```

이미지 최적화에서는 이미지에 다음과 같은 규칙이 적용됩니다.

- Amplify는 GIF, APNG 및 SVG 형식을 최적화하거나 다른 형식으로 변환할 수 없습니다.
- `dangerouslyAllowSVG` 설정을 활성화하지 않으면 SVG 이미지가 제공되지 않습니다.
- 소스 이미지의 너비 또는 높이는 11MB 또는 9,000픽셀을 초과할 수 없습니다.
- 최적화된 이미지의 크기 제한은 4MB입니다.
- HTTPS는 원격 URLs.

HTTP 헤더

Accept 요청 HTTP 헤더는 클라이언트(일반적으로 웹 브라우저)에서 허용되는 이미지 형식(MIME 유형으로 표시됨)을 지정하는 데 사용합니다. 이미지 최적화 서비스에서는 이미지를 지정된 형식으로 변환하려고 시도합니다. 이 헤더에 지정된 값은 형식 쿼리 파라미터보다 우선순위가 높습니다. 예를 들면 Accept 헤더의 유효한 값은 `image/png, image/webp, */*` 입니다. Amplify 배포 매니페스트에 지정된 형식 설정을 통해 목록에 있는 형식으로 제한됩니다. Accept 헤더가 특정 형식을 요청하더라도 해당 형식이 허용 목록에 없으면 요청이 무시됩니다.

URI 요청 파라미터

다음 테이블에서는 이미지 최적화의 URI 요청 파라미터를 설명합니다.

쿼리 파라미터	유형	필수	설명	예제
url	String	예	소스 이미지에 대한 상대 경로 또는 절대 URL입니다. 원격 URL의 경우 https 프로토콜만 지원됩니다. 값은 URL로	`?url=https%3A%2F%2Fwww.example.com%2Fbuffalo.png`

쿼리 파라미터	유형	필수	설명	예제
			인코딩해야 합니다.	
width	숫자	예	최적화된 이미지의 너비입니다(픽셀).	`?width=800`
height	숫자	아니요	최적화된 이미지의 높이입니다(픽셀). 지정하지 않으면 이미지가 너비에 일치하도록 자동으로 규모가 조정됩니다.	`?height=600`
fit	열거형 값: `cover`, `contain`, `fill`, `inside`, `outside`	아니요	지정된 너비와 높이에 알맞게 이미지 크기가 조정되는 방식입니다.	`?width=800&height=600&fit=cover`
position	열거형 값: `center`, `top`, `right`, `bottom`, `left`	아니요	배치가 `cover` 또는 `contain`일 때 사용되는 위치입니다.	`?fit=contain&position=centre`
trim	숫자	아니요	왼쪽 위 픽셀의 지정된 배경색과 비슷한 값이 있는 모든 가장자리에서 픽셀을 잘라냅니다.	`?trim=50`

쿼리 파라미터	유형	필수	설명	예제
확장	객체	아니요	가장 가까운 가장자리 픽셀에서 나온 색상을 사용하여 이미지 가장자리에 픽셀을 추가합니다. 형식은 `{top}_{right}_{bottom}_{left}` 이며 여기서 각 값은 추가할 픽셀의 수입니다.	`?extend=10_0_5_0`
extract	객체	아니요	위쪽, 왼쪽, 너비 및 높이로 구분된 지정된 직사각형으로 이미지를 자릅니다. 형식은 {left}_{top}_{width}_{right}이며 여기서 각 값은 잘라낼 픽셀의 수입니다.	`?extract=10_0_5_0`
형식	String	No	최적화된 이미지에 바람직한 출력 형식입니다.	`?format=webp`
quality	숫자	아니요	이미지 품질입니다(1~100). 이미지 형식을 변환할 때만 사용합니다.	`?quality=50`

쿼리 파라미터	유형	필수	설명	예제
rotate	숫자	아니요	지정된 각도(도)만큼 이미지를 회전합니다.	`?rotate=45`
flip	불린(Boolean)	아니요	이미지를 x축에서 세로(위-아래)로 미러링합니다. 회전이 있는 경우 항상 회전 전에 발생합니다.	`?flip`
flop	불린(Boolean)	아니요	이미지를 y축에서 가로(왼쪽-오른쪽)로 미러링합니다. 회전이 있는 경우 항상 회전 전에 발생합니다.	`?flop`
sharpen	숫자	아니요	선명하게 하면 이미지 가장자리의 선명도가 향상됩니다. 유효한 값은 0.000001~10입니다.	`?sharpen=1`
median	숫자	아니요	중앙값 필터를 적용합니다. 그러면 노이즈가 제거되거나 이미지 가장자리가 부드러워집니다.	`?sharpen=3`

쿼리 파라미터	유형	필수	설명	예제
blur	숫자	아니요	지정된 시그마의 가우스 블러를 적용합니다. 유효한 값은 0.3~1,000입니다.	`?blur=20`
gamma	숫자	아니요	크기가 조정된 이미지에서 인식되는 밝기를 개선하려면 감마 보정을 적용합니다. 값은 1.0~3.0이어야 합니다.	`?gamma=1`
negate	불린(Boolean)	아니요	이미지의 색상을 반전합니다.	`?negate`
normalize	불린(Boolean)	아니요	전체 동적 범위를 포괄하도록 휘도를 늘려 이미지 대비를 개선합니다.	`?normalize`
threshold	숫자	아니요	지정된 임곗값보다 강도가 낮으면 이미지의 모든 픽셀을 검은색 픽셀로 대체합니다. 또는 임곗값보다 높으면 흰색 픽셀로 대체합니다. 유효한 값은 0~255입니다.	`?threshold=155`

쿼리 파라미터	유형	필수	설명	예제
tint	String	No	이미지 휘도를 유지하면서 제공된 RGB를 사용하여 이미지에 색조를 추가합니다.	`?tint=#7743CE`
grayscale	불린(Boolean)	아니요	이미지를 회색조(흑백)로 전환합니다.	`?grayscale`

응답 상태 코드

다음 목록에서는 이미지 최적화의 응답 상태 코드를 설명합니다.

Success - HTTP 상태 코드 200

요청이 이행되었습니다.

BadRequest - HTTP 상태 코드 400

- 입력 쿼리 파라미터가 잘못 지정되었습니다.
- 원격 URL이 `remotePatterns` 설정에 허용된 대로 나열되어 있지 않습니다.
- 원격 URL이 이미지로 확인되지 않습니다.
- 요청된 너비 또는 높이가 `sizes` 설정에 허용된 대로 나열되어 있지 않습니다.
- 요청된 이미지가 SVG인데 `dangerouslyAllowSvg` 설정이 비활성화되어 있습니다.

Not Found - HTTP 상태 코드 404

소스 이미지를 찾을 수 없습니다.

Content too large - HTTP 상태 코드 413

소스 이미지 또는 최적화된 이미지가 허용된 최대 크기(바이트)를 초과합니다.

최적화된 이미지 캐싱 이해

Amplify Hosting에서는 동일한 쿼리 파라미터를 통해 동일한 이미지에 대한 후속 요청이 캐시에서 제공되도록 CDN에 최적화된 이미지를 캐시합니다. 캐시 TTL(Time To Live)은 `Cache-Control` 헤더에 따라 제어됩니다. 다음 목록에서는 `Cache-Control` 헤더 지정 옵션을 설명합니다.

- 이미지 최적화를 대상으로 지정하는 라우팅 규칙 내에서 Cache-Control 키를 사용합니다.
- Amplify 앱에 정의된 사용자 지정 헤더를 사용합니다.
- 원격 이미지의 경우 원격 이미지에서 반환된 Cache-Control 헤더가 적용됩니다.

이미지 최적화 설정에 지정된 minimumCacheTTL에서는 Cache-Control max-age 지시문의 하한을 정의합니다. 예를 들어 원격 이미지 URL에서는 Cache-Control s-max-age=10으로 응답하는데 minimumCacheTTL 값이 60이면 60이 사용됩니다.

모든 SSR 프레임워크에 오픈 소스 어댑터 사용

Amplify Hosting 통합용으로 생성된 모든 SSR 프레임워크 빌드 어댑터를 사용할 수 있습니다. 어댑터가 제공되는 각 프레임워크에 따라 어댑터가 구성되고 빌드 프로세스에 연결되는 방식이 결정됩니다. 일반적으로 npm 개발 종속 항목으로 어댑터를 설치하게 됩니다.

프레임워크로 앱을 만든 후에는 프레임워크 설명서에 따라 Amplify Hosting 어댑터를 설치하고 애플리케이션의 구성 파일에서 어댑터를 구성하는 방법을 알아보세요.

다음으로, 프로젝트의 루트 디렉터리에 amplify.yml 파일을 생성합니다. amplify.yml 파일에서 baseDirectory를 애플리케이션의 빌드 출력 디렉터리에 설정합니다. 프레임워크에서는 빌드 프로세스 중에 어댑터가 실행되어 출력이 Amplify Hosting 배포 번들로 변환됩니다.

빌드 출력 디렉터리의 이름은 무엇이든 상관없지만, .amplify-hosting 파일 이름은 중요합니다. Amplify에서는 먼저 baseDirectory로 정의된 디렉터리를 찾습니다. 디렉터리가 있으면 Amplify에서는 거기에서 빌드 출력을 찾습니다. 디렉터리가 없으면 고객이 정의하지 않았더라도 Amplify가 .amplify-hosting 내부에서 빌드 출력을 찾습니다.

다음은 앱의 빌드 설정 예시입니다. 빌드 출력이 .amplify-hosting 폴더에 있다는 것을 나타내도록 baseDirectory가 .amplify-hosting으로 설정되어 있습니다. .amplify-hosting 폴더의 콘텐츠가 Amplify Hosting 배포 사양과 일치하기만 하면 앱이 배포됩니다.

```
version: 1
frontend:
  preBuild:
    commands:
      - npm install
  build:
    commands:
      - npm run build
  artifacts:
```

```
    baseDirectory: .amplify-hosting
```

프레임워크 어댑터가 사용되도록 앱을 구성한 후 Amplify Hosting에 배포할 수 있습니다. 자세한 지침은 Amplify에 SSR 앱 배포 단원을 참조하십시오.

Amazon S3 버킷에서 Amplify에 정적 웹 사이트 배포

Amplify Hosting과 Amazon S3 간 통합을 사용하여 몇 번의 클릭으로 S3에 저장된 정적 웹 사이트 콘텐츠를 호스팅할 수 있습니다. Amplify Hosting에 배포하면 다음과 같은 이점 및 기능을 확보할 수 있습니다.

- CloudFront로 구동되는 전역적으로 사용 가능한 AWS 콘텐츠 전송 네트워크(CDN)에 자동 배포
- HTTPS 지원
- Amplify 콘솔을 사용하여 웹 사이트를 사용자 지정 도메인에 간편하게 연결
- 기존 사용자 지정 SSL 인증서 사용
- 내장된 액세스 로그 및 CloudWatch 지표로 웹 사이트 모니터링
- 웹 사이트에서 암호 보호 설정
- Amplify 콘솔에서 리디렉션 및 다시 쓰기 규칙 생성

Amplify 콘솔, AWS CLI또는 AWS SDKs. 자신의 계정에 있는 Amazon S3 범용 버킷에서만 Amplify에 배포할 수 있습니다. Amplify는 교차 계정 S3 버킷 액세스를 지원하지 않습니다.

Amazon S3 범용 버킷에서 Amplify Hosting으로 애플리케이션을 배포하는 경우 AWS 요금은 Amplify의 요금 모델을 기반으로 합니다. 자세한 내용은 AWS Amplify 요금을 참조하세요.

> **Important**
>
> Amazon S3를 사용할 수 AWS 리전 있는 모든에서 Amplify Hosting을 사용할 수 있는 것은 아닙니다. 정적 웹 사이트를 Amplify Hosting에 배포하려면 웹 사이트가 저장된 Amazon S3 범용 버킷이 Amplify를 사용할 수 있는 리전에 있어야 합니다. Amplify를 사용할 수 있는 리전 목록은 Amazon Web Services 일반 참조의 Amplify 엔드포인트를 참조하세요.

Amazon S3에서 Amplify Hosting으로 정적 웹 사이트를 배포하고 업데이트하는 방법을 알아보려면 다음 주제를 참조하세요.

주제

- Amplify 콘솔을 사용하여 S3에서 정적 웹 사이트 배포
- AWS SDKs를 S3 사용하여에서 정적 웹 사이트를 배포하는 버킷 정책 생성
- S3 버킷에서 Amplify로 배포된 정적 웹 사이트 업데이트

- .zip 파일 대신 버킷 및 접두사를 사용하도록 S3 배포 업데이트

Amplify 콘솔을 사용하여 S3에서 정적 웹 사이트 배포

Amplify 콘솔을 사용하여 Amazon S3 범용 버킷에서 새 정적 웹 사이트를 배포하려면 다음 지침을 따릅니다.

Amplify 콘솔을 사용하여 Amazon S3 범용 버킷에서 정적 웹 사이트를 배포하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/amplify/ Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 없이 배포를 선택합니다.
4. Next(다음)를 선택합니다.
5. 수동 배포 시작 페이지에서 다음을 수행합니다.
   a. API 이름에 앱의 이름을 입력합니다.
   b. 브랜치 이름에 배포할 브랜치의 이름을 입력합니다.
6. 방법에서 Amazon S3를 선택합니다.
7. 호스팅할 객체의 S3 위치에서 찾아보기를 선택합니다. 사용할 Amazon S3 범용 버킷을 선택한 다음 접두사 선택을 선택합니다.
8. 저장 및 배포를 선택합니다.

AWS SDKs를 S3 사용하여에서 정적 웹 사이트를 배포하는 버킷 정책 생성

AWS SDKs를 사용하여 Amazon S3에서 Amplify Hosting으로 정적 웹 사이트를 배포할 수 있습니다. SDK를 사용하여 웹 사이트를 배포하는 경우 Amplify Hosting에 S3 버킷에서 객체를 검색할 수 있는 권한을 부여하는 자체 버킷 정책을 생성해야 합니다.

버킷 정책을 생성하는 방법에 대해 자세히 알아보려면 Amazon Simple Storage Service 사용 설명서의 Amazon S3의 버킷 정책을 참조하세요.

다음 예제 버킷 정책은 지정된 Amplify 애플리케이션 ID 및 브랜치에 대해 버킷을 나열하고 버킷 객체를 검색할 수 있는 권한을 AWS 계정 Amplify Hosting에 부여합니다.

이 예제 사용

- *amzn-s3-demo-website-bucket/prefix*를 웹 사이트 버킷 및 접두사 이름으로 바꿉니다.
- *111122223333*을 AWS 계정 ID로 바꿉니다.
- *region-id*를와 같이 Amplify 애플리케이션이 AWS 리전 있는 로 바꿉니다**us-east-1**.
- *app_id*를 Amplify 애플리케이션 ID로 바꿉니다. 이 정보는 Amplify 콘솔에서 확인할 수 있습니다.
- *branch_name*을 브랜치 이름으로 바꿉니다.

> **Note**
>
> 버킷 정책에서 `aws:SourceArn`은 URL 인코딩(백분율 인코딩) 브랜치 ARN이어야 합니다.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "AllowAmplifyToListPrefix_appid_branch_prefix_",
            "Effect": "Allow",
            "Principal": {
                "Service": "amplify.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-website-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn%3Aaws%3Aamplify%3Aregion-
id%3A111122223333%3Aapps%2Fapp_id%2Fbranches%2Fbranch_name",
                    "s3:prefix": ""
                }
            }
        },
        {
            "Sid": "AllowAmplifyToReadPrefix__appid_branch_prefix_",
            "Effect": "Allow",
            "Principal": {
                "Service": "amplify.amazonaws.com"
            },
            "Action": "s3:GetObject",
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-website-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn%3Aaws%3Aamplify%3Aregion-
id%3A111122223333%3Aapps%2Fapp_id%2Fbranches%2Fbranch_name"
                }
            }
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::amzn-s3-demo-website-bucket/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

S3 버킷에서 Amplify로 배포된 정적 웹 사이트 업데이트

Amplify에서 호스팅되는 범용 S3 버킷에서 정적 웹 사이트의 객체를 업데이트하는 경우 변경 사항이 적용되도록 애플리케이션을 Amplify Hosting에 재배포해야 합니다. Amplify Hosting은 S3 버킷의 변경 사항을 자동으로 감지하지 않습니다. AWS Command Line Interface (CLI)를 사용하여 웹 사이트를 업데이트하는 것이 좋습니다.

S3에 업데이트 동기화

웹 사이트의 프로젝트 파일을 변경한 후 다음 s3 sync 명령을 사용하여 로컬 소스 디렉터리에 대한 변경 사항을 대상 Amazon S3 범용 버킷과 동기화합니다. 이 예제를 사용하려면 *<source>*를 로컬 디렉터리 이름으로 바꾸고 *<target>*을 Amazon S3 버킷 이름으로 바꿉니다.

```
aws s3 sync <source> <target>
```

Amplify Hosting에 웹 사이트 재배포

다음 amplify start-deployment 명령을 사용하여 업데이트된 애플리케이션을 Amazon S3 버킷에서 Amplify Hosting으로 재배포합니다. 이 예제를 사용하려면 *<app_id>*를 Amplify 애플리케이션의 ID로, *<branch_name>*을 브랜치 이름으로, *s3://amzn-s3-demo-website-bucket/prefix*를 S3 버킷 및 접두사로 바꿉니다.

```
aws amplify start-deployment --app-id <app_id> --branch-name <branch_name> --source-
url s3://amzn-s3-demo-website-bucket/prefix --source-url-type BUCKET_PREFIX
```

.zip 파일 대신 버킷 및 접두사를 사용하도록 S3 배포 업데이트

Amazon S3 범용 버킷의 .zip 파일에서 Amplify Hosting에 배포된 기존 정적 웹 사이트가 이미 있는 경우 호스팅할 객체가 저장된 버킷 이름 및 접두사를 사용하도록 애플리케이션 배포를 업데이트할 수 있습니다. 이러한 유형의 배포를 사용하면 빌드 출력의 압축된 콘텐츠가 포함된 별도의 파일을 버킷에 업로드할 필요가 없습니다.

정적 웹 사이트를 .zip 파일에서 버킷 콘텐츠로 마이그레이션하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/amplify/ Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 .zip 파일 사용에서 애플리케이션 파일 직접 사용으로 마이그레이션하려는 수동으로 배포된 앱의 이름을 선택합니다.
3. 애플리케이션의 개요 페이지에서 업데이트 배포를 선택합니다.
4. 업데이트 배포 페이지의 방법에서 Amazon S3를 선택합니다.
5. 호스팅할 객체의 S3 위치에서 찾아보기를 선택합니다. 사용할 버킷을 선택한 다음 접두사 선택을 선택합니다.
6. 저장 및 배포를 선택합니다.

Git 리포지토리 없이 Amplify에 애플리케이션 배포

수동 배포를 사용하면 Git 공급자를 연결하지 않고도 Amplify Hosting으로 웹 앱을 게시할 수 있습니다. 데스크톱에서 압축 폴더를 끌어서 놓고 몇 초 만에 사이트를 호스팅할 수 있습니다. 또는 Amazon S3 버킷의 자산을 참조하거나 파일이 저장된 위치의 퍼블릭 URL을 지정할 수 있습니다.

> **Note**
>
> Amazon S3 복사 작업 제약으로 인해 수동 배포의 최대 .zip 파일 크기 제한은 5GB입니다. 빌드 아티팩트 중 하나라도이 크기를 초과하는 경우 이를 더 작은 아카이브로 나누거나 대체 배포 방법을 사용하는 것이 좋습니다.

Amazon S3의 경우 새 자산이 업로드될 때마다 사이트를 업데이트하도록 AWS Lambda 트리거를 설정할 수도 있습니다. 이 시나리오 설정에 대한 자세한 내용은 Amazon S3, Dropbox 또는 데스크톱에 저장된 파일을 AWS Amplify 콘솔에 배포하기 블로그 게시물을 참조하십시오.

Amplify Hosting은 서버 측 렌더링(SSR) 앱의 수동 배포를 지원하지 않습니다. 자세한 내용은 Amplify Hosting을 통해 서버 측 렌더링 애플리케이션 배포 단원을 참조하십시오.

수동 배포 끌어서 놓기

드래그 앤 드롭을 사용하여 앱을 수동으로 배포하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 오른쪽 상단 모서리에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 없이 배포를 선택합니다. 그리고 다음을 선택합니다.
4. 수동 배포 시작 페이지에서 앱 이름에 앱 이름을 입력합니다.
5. 브랜치 이름에 **`development`** 또는 **`production`**과 같은 의미 있는 이름을 입력합니다.
6. 방법에서 드래그 앤 드롭을 선택합니다.
7. 데스크톱에서 드롭 영역으로 파일을 끌어 놓거나 .zip 폴더 선택을 사용하여 컴퓨터에서 파일을 선택합니다. 끌어 놓거나 선택하는 파일은 빌드 출력의 내용이 포함된 압축 폴더여야 합니다.
8. 저장 및 배포를 선택합니다.

Amazon S3 또는 URL 수동 배포

> **Note**
>
> S3에서 정적 웹 사이트를 배포하는 경우 다음 절차에 따라 빌드 출력 내용이 포함된 압축 폴더를 S3 버킷에 업로드해야 합니다. 버킷 이름 및 접두사를 사용하여 S3에서 직접 정적 웹 사이트를 배포하는 것이 좋습니다. 이 간소화된 프로세스에 대한 자세한 내용은 Amazon S3 버킷에서 Amplify에 정적 웹 사이트 배포 섹션을 참조하세요.

Amazon S3 또는 퍼블릭 URL에서 앱을 수동으로 배포하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 오른쪽 상단 모서리에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 없이 배포를 선택합니다. 그리고 다음을 선택합니다.
4. 수동 배포 시작 페이지에서 앱 이름에 앱 이름을 입력합니다.
5. 브랜치 이름에 **`development`** 또는 **`production`**과 같은 의미 있는 이름을 입력합니다.
6. 방법에서 Amazon S3 또는 모든 URL을 선택합니다.
7. 파일 업로드 절차는 업로드 방법에 따라 다릅니다.
   - Amazon S3
     a. S3 location of objects to host에서 S3 찾아보기를 선택합니다. 그런 다음 목록에서 Amazon S3 버킷의 이름을 선택합니다. 선택한 버킷에 액세스 제어 목록(ACL)을 활성화해야 합니다. 자세한 내용은 수동 배포를 위한 Amazon S3 버킷 액세스 문제 해결 단원을 참조하십시오.
     b. 배포할 zip 파일의 이름을 선택합니다.
     c. 접두사 선택을 선택합니다.
   - 모든 URL
     - 리소스 URL에서 배포할 .zip 파일의 URL을 입력합니다.
8. 저장 및 배포를 선택합니다.

> **Note**
>
> 압축 폴더를 만들 때는 최상위 폴더가 아닌 빌드 출력의 콘텐츠를 압축해야 합니다. 예를 들어 빌드 출력에서 이름이 “build” 또는 “public”인 폴더가 생성되면 먼저 해당 폴더로 이동하여 모든 콘텐츠를 선택한 다음 압축합니다. 이렇게 하지 않으면 사이트의 루트 디렉터리가 제대로 초기화되지 않기 때문에 “액세스 거부됨” 오류가 표시됩니다.

수동 배포를 위한 Amazon S3 버킷 액세스 문제 해결

Amazon S3 버킷 생성 시 Amazon S3 객체 소유권 설정을 사용하여 버킷에 대한 액세스 제어 목록(ACL) 활성화 여부를 제어합니다. Amazon S3 버킷에서 Amplify에 앱을 수동으로 배포하려면 버킷에 ACL을 활성화해야 합니다.

Amazon S3 버킷에서 배포 시 `AccessControlList` 오류가 발생하는 경우, ACL이 비활성화된 상태로 버킷이 생성되었으므로 Amazon S3 콘솔에서 ACL을 활성화해야 합니다. 지침을 보려면 Amazon Simple Storage Service 사용 설명서에서 기존 버킷에 객체 소유권 설정을 참조하십시오.

Amplify 애플리케이션의 빌드 구성 관리

Amplify 배포에 대한 빌드 설정 및 구성을 사용자 지정할 수 있습니다. 애플리케이션을 배포하면 Amplify가 프런트엔드 프레임워크 및 관련 빌드 설정을 자동으로 감지합니다. 애플리케이션의 빌드 사양(buildspec)에서 빌드 설정을 사용자 지정하여 환경 변수를 추가하고, 빌드 명령을 실행하고, 빌드 종속성을 지정할 수 있습니다.

Amplify의 기본 빌드 이미지에는 여러 패키지 및 종속성이 사전 설치되어 있지만 라이브 패키지 업데이트 기능을 사용하여 특정 버전을 지정하거나 최신 버전이 항상 설치되도록 할 수도 있습니다. Amplify의 기본 컨테이너를 사용하여 빌드 중에 설치하는 데 시간이 오래 걸리는 특정 종속성이 있는 경우 사용자 지정 빌드 이미지를 생성할 수 있습니다. 빌드 인스턴스 크기를 사용자 지정하여 애플리케이션 배포에 필요한 CPU, 메모리 및 디스크 공간 리소스를 제공할 수도 있습니다.

빌드는 Git 리포지토리에 대한 각 커밋과 새 배포마다 자동으로 시작됩니다. 수신 웹후크 기능을 설정하여 Git 리포지토리에 커밋하지 않고 빌드를 시작할 수 있습니다.

빌드 알림 기능을 사용하면 빌드 성공 및 실패에 대한 정보를 팀원과 공유할 수 있습니다.

주제

Amplify 애플리케이션의 빌드 설정 구성

애플리케이션을 배포하면 Amplify는 Git 리포지토리에서 앱의 `package.json` 파일을 검사하여 프론트엔드 프레임워크 및 관련 빌드 설정을 자동으로 감지합니다. 앱의 빌드 설정을 저장하는 옵션은 다음과 같습니다.

- Amplify 콘솔에 빌드 설정 저장 - Amplify 콘솔은 빌드 설정을 자동 감지하여 Amplify 콘솔을 통해 액세스할 수 있도록 저장합니다. Amplify는 리포지토리에 저장된 `amplify.yml` 파일이 없는 한 모든 브랜치에 이러한 설정을 적용합니다.
- 리포지토리에 빌드 설정 저장 - `amplify.yml` 파일을 다운로드하여 리포지토리의 루트에 추가합니다.

> **Note**
>
> 앱이 연속 배포되도록 설정하고 git 리포지토리에 연결된 경우에만 Amplify 콘솔의 호스팅 메뉴에 빌드 설정이 표시됩니다. 이러한 유형의 배포에 대한 지침은 시작하기를 참조하세요.

빌드 사양 참조

Amplify 애플리케이션의 빌드 사양(buildspec)은 Amplify가 빌드를 실행하는 데 사용하는 YAML 설정 및 빌드 명령의 모음입니다. 다음 목록은 이러한 설정과 사용 방법을 설명합니다.

version

Amplify YAML 버전 번호입니다.

appRoot

이 애플리케이션이 있는 리포지토리 내 경로입니다. 여러 개의 애플리케이션이 정의되어 있지 않으면 무시됩니다.

env

이 섹션에 환경 변수를 추가합니다. 콘솔을 사용하여 환경 변수를 추가할 수도 있습니다.

백엔드

Amplify CLI 명령을 실행하여 백엔드 프로비저닝, Lambda 함수 업데이트 또는 GraphQL 스키마를 연속 배포의 일환으로 수행합니다.

프런트엔드

프런트엔드 빌드 명령을 실행합니다.

테스트

테스트 단계에서 명령을 실행합니다. 앱에 테스트를 추가하는 방법에 대해 알아보십시오.

빌드 단계

프런트엔드와 백엔드, 테스트에는 빌드의 각 시퀀스 동안 실행되는 명령을 나타내는 세 가지 단계가 있습니다.

- preBuild - preBuild 스크립트는 실제 빌드가 시작되기 전에 실행되지만 Amplify가 종속성을 설치한 후에 실행됩니다.
- build - 사용자의 빌드 명령입니다.

- postBuild - 사후 빌드 스크립트는 빌드가 종료되고 Amplify가 필요한 모든 결과물을 출력 디렉터리에 복사한 후 실행됩니다.

buildpath

빌드를 실행하는 데 사용할 경로입니다. Amplify는 이 경로를 사용하여 빌드 아티팩트를 찾습니다. 경로를 지정하지 않으면 Amplify는 모노레포 앱 루트를 사용합니다(예: apps/app).

artifacts>base-directory

빌드 아티팩트가 있는 디렉터리입니다.

artifacts>files

배포하려는 아티팩트의 파일을 지정합니다. 모든 파일을 포함하려면 **/*를 입력합니다.

cache

node_modules 폴더와 같은 빌드 시간 종속성을 지정합니다. 첫 번째 빌드 중에 여기에 제공된 경로가 캐시됩니다. 후속 빌드에서 Amplify는 명령을 실행하기 전에 캐시를 동일한 경로로 복원합니다.

Amplify는 제공된 모든 캐시 경로를 프로젝트 루트에 상대적인 것으로 간주합니다. 그러나 Amplify는 프로젝트 루트 외부로의 통과를 허용하지 않습니다. 예를 들어 절대 경로를 지정하면 오류 없이 빌드가 성공하지만 경로는 캐시되지 않습니다.

빌드 사양 YAML 구문 참조

다음의 빌드 사양 예제는 기본 YAML 구문을 나타냅니다.

```
version: 1
env:
  variables:
    key: value
backend:
  phases:
    preBuild:
      commands:
        - *enter command*
    build:
      commands:
        - *enter command*
    postBuild:
        commands:
        - *enter command*
```

```
frontend:
  buildpath:
  phases:
    preBuild:
      commands:
        - cd react-app
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    files:
        - location
        - location
    discard-paths: yes
    baseDirectory: location
  cache:
    paths:
        - path # A cache path relative to the project root
        - path # Traversing outside of the project root is not allowed
test:
  phases:
    preTest:
      commands:
        - *enter command*
    test:
      commands:
        - *enter command*
    postTest:
      commands:
        - *enter command*
  artifacts:
    files:
        - location
        - location
    configFilePath: *location*
    baseDirectory: *location*
```

빌드 사양 편집

Amplify 콘솔에서 빌드 사양(buildspec)을 편집하여 애플리케이션의 빌드 설정을 사용자 지정할 수 있습니다. 이러한 빌드 설정은 Git 리포지토리에 `amplify.yml` 파일이 저장된 브랜치를 제외하고 앱의 모든 브랜치에 적용됩니다.

Amplify 콘솔에서 빌드 설정을 편집하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 빌드 설정을 편집하려는 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 빌드 설정을 선택합니다.
4. 빌드 설정 페이지의 앱 빌드 사양 섹션에서 편집을 선택합니다.
5. 빌드 사양 편집 창에서 업데이트를 입력합니다.
6. 저장을 선택합니다.

다음 항목에 설명된 예제를 사용하여 특정 시나리오에 대한 빌드 설정을 업데이트할 수 있습니다.

주제

- 스크립팅을 사용하여 브랜치별 빌드 설정
- 하위 폴더로 이동하기 위한 명령 설정
- Gen 1 앱의 프론트엔드를 사용하여 백엔드 배포
- 출력 폴더 설정
- 빌드의 일부로 패키지 설치
- 프라이빗 npm 레지스트리 사용
- OS 패키지 설치
- 모든 빌드에 대한 키-값 스토리지 설정
- 커밋을 위한 빌드 건너뛰기
- 모든 커밋에서 자동 빌드 끄기
- diff 기반 프론트엔드 빌드 및 배포 구성
- Gen 1 앱을 위한 diff 기반 백엔드 빌드 구성

스크립팅을 사용하여 브랜치별 빌드 설정

배시 쉘 스크립팅을 사용하여 브랜치별 빌드 설정을 지정할 수 있습니다. 예를 들어 다음 스크립트는 시스템 환경 변수 $AWS_BRANCH를 사용하여 브랜치 이름이 main인 경우 한 세트의 명령을 실행하고, 브랜치 이름이 dev인 경우 다른 세트의 명령을 실행합니다.

```
frontend:
  phases:
```

```
    build:
      commands:
        - if [ "${AWS_BRANCH}" = "main" ]; then echo "main branch"; fi
        - if [ "${AWS_BRANCH}" = "dev" ]; then echo "dev branch"; fi
```

하위 폴더로 이동하기 위한 명령 설정

모노레포의 경우 사용자는 빌드를 실행하기 위해 폴더로 연속적으로 cd할 수 있기를 원합니다. cd 명령을 실행한 후에는 빌드의 모든 스테이지에 적용되므로 별도의 단계에서 명령을 반복하지 않아도 됩니다.

```
version: 1
env:
  variables:
    key: value
frontend:
  phases:
    preBuild:
      commands:
        - cd react-app
        - npm ci
    build:
      commands:
        - npm run build
```

Gen 1 앱의 프론트엔드를 사용하여 백엔드 배포

> **Note**
>
> 이 섹션은 Amplify Gen 1 애플리케이션에만 적용됩니다. Gen 1 백엔드는 Amplify Studio 및 Amplify 명령줄 인터페이스(CLI)를 사용하여 생성합니다.

amplifyPush 명령은 백엔드 배포에 도움을 제공하는 도우미 스크립트입니다. 아래의 빌드 설정은 현재 브랜치에 대해 배포할 올바른 백엔드 환경을 자동으로 결정합니다.

```
version: 1
env:
  variables:
    key: value
backend:
```

```
  phases:
    build:
      commands:
        - amplifyPush --simple
```

출력 폴더 설정

다음 빌드 설정에서는 출력 디렉터리를 퍼블릭 폴더로 설정합니다.

```
frontend:
  phases:
    commands:
      build:
        - yarn run build
  artifacts:
    baseDirectory: public
```

빌드의 일부로 패키지 설치

빌드하는 동안 npm 또는 yarn 명령을 사용하여 패키지를 설치할 수 있습니다.

```
frontend:
  phases:
    build:
      commands:
        - npm install -g <package>
        - <package> deploy
        - yarn run build
  artifacts:
    baseDirectory: public
```

프라이빗 npm 레지스트리 사용

빌드 설정에서 프라이빗 레지스트리에 대한 참조를 추가하거나 환경 변수로 추가할 수 있습니다.

```
build:
  phases:
    preBuild:
      commands:
        - npm config set <key> <value>
        - npm config set registry https://registry.npmjs.org
```

```
        - npm config set always-auth true
        - npm config set email hello@amplifyapp.com
        - yarn install
```

OS 패키지 설치

Amplify의 AL2023 이미지는 이름이 amplify인 권한이 없는 사용자를 사용하여 코드를 실행합니다. Amplify는 이 사용자에게 Linux sudo 명령을 사용하여 OS 명령을 실행할 수 있는 권한을 부여합니다. 누락된 종속성에 OS 패키지를 설치하려는 경우 yum 및 rpm과 같은 명령을 sudo와 함께 사용할 수 있습니다.

다음 예제 빌드 섹션에서는 sudo 명령을 사용하여 OS 패키지를 설치하기 위한 구문을 보여줍니다.

```
build:
  phases:
    preBuild:
      commands:
        - sudo yum install -y <package>
```

모든 빌드에 대한 키-값 스토리지 설정

envCache은(는) 빌드 시 키-값 스토리지를 제공합니다 envCache에 저장된 값은 빌드 중에만 수정할 수 있으며 다음 빌드에서 재사용할 수 있습니다. envCache을(를) 사용하여 배포된 환경에 정보를 저장하고 연속 빌드에서 빌드 컨테이너에 사용할 수 있습니다. envCache에 저장된 값과 달리, 빌드 중 환경 변수의 변경 사항은 미래 빌드에 지속되지 않습니다.

사용 예:

```
envCache --set <key> <value>
envCache --get <key>
```

커밋을 위한 빌드 건너뛰기

특정 커밋에서 자동 빌드를 건너뛰려면 커밋 메시지 끝에 [skip-cd] 텍스트를 포함합니다.

모든 커밋에서 자동 빌드 끄기

모든 코드 커밋에 대해 자동 빌드를 끄도록 Amplify를 구성할 수 있습니다. 설정하려면 앱 설정, 브랜치 설정을 선택한 다음 연결된 브랜치가 있는 브랜치 섹션을 찾습니다. 브랜치를 선택한 다음 작업, 자동 빌드 비활성화를 선택합니다. 해당 브랜치에 대한 새 커밋은 더 이상 새 빌드를 트리거하지 않습니다.

diff 기반 프론트엔드 빌드 및 배포 구성

diff 기반 프런트엔드 빌드를 사용하도록 Amplify를 구성할 수 있습니다. 활성화 시 각 빌드를 시작할 때 Amplify가 기본적으로 사용자의 `appRoot` 또는 `/src/` 폴더에서 diff 실행을 시도합니다. Amplify가 차이점을 발견하지 못하면 프런트엔드 빌드, 테스트(구성된 경우) 및 배포 단계를 건너뛰고 호스팅된 앱을 업데이트하지 않습니다.

diff 기반 프런트엔드 빌드 및 배포를 구성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. diff 기반 프런트엔드 빌드를 구성하고 배포할 앱을 선택합니다.
3. 탐색 창에서 호스팅, 환경 변수를 선택합니다.
4. 환경 변수 섹션에서 변수 관리를 선택합니다.
5. 환경 변수를 구성하는 절차는 diff 기반 프런트엔드 빌드 및 배포의 활성화 여부에 따라 달라집니다.
   - diff 기반 프런트엔드 빌드 및 배포를 활성화하려면
     a. 변수 관리 섹션의 변수에 `AMPLIFY_DIFF_DEPLOY`를 입력합니다.
     b. 값에 `true`를 입력합니다.
   - diff 기반 프런트엔드 빌드 및 배포를 비활성화하려면
     - 다음 중 하나를 수행하십시오.
       - 변수 관리 섹션에서 `AMPLIFY_DIFF_DEPLOY`을(를) 찾습니다. 값에 `false`를 입력합니다.
       - `AMPLIFY_DIFF_DEPLOY` 환경 변수를 제거합니다.
6. 저장을 선택합니다.

선택적으로 `AMPLIFY_DIFF_DEPLOY_ROOT` 환경 변수를 설정하여 기본 경로를 리포지토리의 루트와 관련된 경로(예: `dist`)로 재정의할 수 있습니다.

Gen 1 앱을 위한 diff 기반 백엔드 빌드 구성

> **Note**
>
> 이 섹션은 Amplify Gen 1 애플리케이션에만 적용됩니다. Gen 1 백엔드는 Amplify Studio 및 Amplify 명령줄 인터페이스(CLI)를 사용하여 생성합니다.

AMPLIFY_DIFF_BACKEND 환경 변수를 사용하여 diff 기반 백엔드 빌드를 사용하도록 Amplify Hosting을 구성할 수 있습니다. diff 기반 백엔드 빌드를 활성화하면 각 빌드를 시작할 때 Amplify가 리포지토리의 amplify 폴더에서 diff 실행을 시도합니다. Amplify에서 차이점을 발견하지 못하면 백엔드 빌드 단계를 건너뛰고 백엔드 리소스를 업데이트하지 않습니다. 프로젝트의 리포지토리에 amplify 폴더가 없는 경우 Amplify는 AMPLIFY_DIFF_BACKEND 환경 변수 값을 무시합니다.

현재 백엔드 단계의 빌드 설정에 사용자 지정 명령이 지정되어 있는 경우, 조건부 백엔드 빌드는 작동하지 않습니다. 이러한 사용자 지정 명령을 실행하려면 앱의 amplify.yml 파일에 있는 빌드 설정의 프런트엔드 단계로 해당 명령을 이동해야 합니다.

diff 기반 백엔드 빌드를 구성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. diff 기반 백엔드 빌드를 구성할 앱을 선택합니다.
3. 탐색 창에서 호스팅, 환경 변수를 선택합니다.
4. 환경 변수 섹션에서 변수 관리를 선택합니다.
5. 환경 변수를 구성하는 절차는 diff 기반 백엔드 빌드의 활성화 여부에 따라 달라집니다.
   - diff 기반 백엔드 빌드를 활성화하려면
     a. 변수 관리 섹션의 변수에 AMPLIFY_DIFF_BACKEND를 입력합니다.
     b. 값에 true를 입력합니다.
   - diff 기반 백엔드 빌드를 비활성화하려면
     - 다음 중 하나를 수행하십시오.
       - 변수 관리 섹션에서 AMPLIFY_DIFF_BACKEND을(를) 찾습니다. 값에 false를 입력합니다.
       - AMPLIFY_DIFF_BACKEND 환경 변수를 제거합니다.
6. 저장을 선택합니다.

모노 리포지토리 빌드 설정 구성

단일 리포지토리에 여러 프로젝트 또는 마이크로서비스를 저장하는 것을 모노레포라고 합니다. Amplify Hosting을 사용하면 여러 빌드 구성 또는 브랜치 구성을 만들지 않고도 애플리케이션을 모노레포로 배포할 수 있습니다.

Amplify는 일반 모노레포 앱뿐만 아니라 npm 워크스페이스, pnpm 워크스페이스, Yarn 워크스페이스, Nx 및 Turborepo를 사용하여 생성된 모노레포 앱을 지원합니다. 앱 배포 시 Amplify는 사용 중인 모노레포 빌드 도구를 자동으로 감지합니다. Amplify는 npm 워크스페이스, Yarn 워크스페이스 또는 Nx의 앱에 대한 빌드 설정을 자동으로 적용합니다. Turborepo와 pnpm 앱에는 추가 구성이 필요합니다. 자세한 내용은 Turborepo 및 pnpm 모노레포 앱 구성 단원을 참조하십시오.

Amplify 콘솔에서 모노레포에 대한 빌드 설정을 저장하거나 `amplify.yml` 파일을 다운로드하여 리포지토리의 루트에 추가할 수 있습니다. Amplify는 리포지토리에서 `amplify.yml` 파일을 찾지 않는 한, 콘솔에 저장된 설정을 모든 브랜치에 적용합니다. `amplify.yml` 파일이 있는 경우 해당 설정은 Amplify 콘솔에 저장된 모든 빌드 설정을 재정의합니다.

모노 리포지토리 빌드 사양 YAML 구문 참조

모노레포 빌드 사양의 YAML 구문은 단일 애플리케이션이 포함된 리포지토리의 YAML 구문과 다릅니다. 모노레포의 경우 애플리케이션 목록에서 각 프로젝트를 선언합니다. 모노레포 빌드 사양에 선언하는 각 애플리케이션에 대해 다음과 같은 추가 `appRoot` 키를 제공해야 합니다.

appRoot

애플리케이션이 시작되는 리포지토리 내의 루트입니다. 이 키는 반드시 존재해야 하며 `AMPLIFY_MONOREPO_APP_ROOT` 환경 변수와 동일한 값을 가져야 합니다. 이 환경 변수 설정에 대한 지침은 AMPLIFY_MONOREPO_APP_ROOT 환경 변수 설정 섹션을 참조하십시오.

다음 모노레포 빌드 사양 예제는 동일한 리포지토리에서 여러 Amplify 애플리케이션을 선언하는 방법을 설명합니다. 두 앱 `react-app`, `angular-app`은 `applications` 목록에 선언되어 있습니다. 각 앱의 `appRoot` 키는 앱이 리포지토리의 `apps` 루트 폴더에 있음을 나타냅니다.

`buildpath` 속성은 모노레포 프로젝트 루트에서 앱을 실행하고 빌드하도록 `/`로 설정되어 있습니다. `baseDirectory` 속성은의 상대 경로입니다`buildpath`.

모노레포 빌드 사양 YAML 구문

```
version: 1
applications:
  - appRoot: apps/react-app
    env:
      variables:
        key: value
    backend:
      phases:
        preBuild:
```

```
        commands:
          - *enter command*
      build:
        commands:
          - *enter command*
      postBuild:
          commands:
          - *enter command*
  frontend:
    buildPath: / # Run install and build from the monorepo project root
    phases:
      preBuild:
        commands:
          - *enter command*
          - *enter command*
      build:
        commands:
          - *enter command*
    artifacts:
      files:
          - location
          - location
      discard-paths: yes
      baseDirectory: location
    cache:
      paths:
          - path
          - path
  test:
    phases:
      preTest:
        commands:
          - *enter command*
      test:
        commands:
          - *enter command*
      postTest:
        commands:
          - *enter command*
    artifacts:
      files:
          - location
          - location
      configFilePath: *location*
```

```
      baseDirectory: *location*
  - appRoot: apps/angular-app
    env:
      variables:
        key: value
    backend:
      phases:
        preBuild:
          commands:
            - *enter command*
        build:
          commands:
            - *enter command*
        postBuild:
            commands:
            - *enter command*
    frontend:
      phases:
        preBuild:
          commands:
            - *enter command*
            - *enter command*
        build:
          commands:
            - *enter command*
      artifacts:
        files:
            - location
            - location
        discard-paths: yes
        baseDirectory: location
      cache:
        paths:
            - path
            - path
    test:
      phases:
        preTest:
          commands:
            - *enter command*
        test:
          commands:
            - *enter command*
        postTest:
```

```
          commands:
            - *enter command*
      artifacts:
        files:
            - location
            - location
        configFilePath: *location*
        baseDirectory: *location*
```

다음 예제 빌드 사양을 사용하는 앱은 프로젝트 루트 아래에 빌드되고 빌드 아티팩트는에 있습니다/packages/nextjs-app/.next.

```
applications:
  - frontend:
      buildPath: '/'  # run install and build from monorepo project root
      phases:
        preBuild:
          commands:
            - npm install
        build:
          commands:
            - npm run build --workspace=nextjs-app
      artifacts:
        baseDirectory: packages/nextjs-app/.next
        files:
          - '**/*'
      cache:
        paths:
          - node_modules/**/*
    appRoot: packages/nextjs-app
```

AMPLIFY_MONOREPO_APP_ROOT 환경 변수 설정

모노레포에 저장된 앱을 배포하는 경우, 앱의 AMPLIFY_MONOREPO_APP_ROOT 환경 변수는 리포지토리의 루트를 기준으로 앱 루트 경로와 동일한 값을 가져야 합니다. 예를 들어, 이름이 apps으로 지정된 루트 폴더가 있고 app1, app2, app3을 포함하는 ExampleMonorepo라는 이름의 모노레포는 다음 디렉터리 구조를 가집니다.

```
ExampleMonorepo
  apps
    app1
```

```
    app2
    app3
```

이 예제에서 app1에 대한 AMPLIFY_MONOREPO_APP_ROOT 환경 변수의 값은 apps/app1입니다.

Amplify 콘솔을 사용하여 모노레포 앱을 배포하면 콘솔은 앱의 루트 경로에 지정한 값을 사용하여 AMPLIFY_MONOREPO_APP_ROOT 환경 변수를 자동으로 설정합니다. 그러나 모노 리포지토리 앱이 이미 Amplify에 있거나를 사용하여 배포된 경우 Amplify 콘솔의 AMPLIFY_MONOREPO_APP_ROOT 환경 변수 섹션에서 환경 변수를 수동으로 설정해야 AWS CloudFormation합니다.

배포 중 AMPLIFY_MONOREPO_APP_ROOT 환경 변수 자동 설정

다음 지침은 Amplify 콘솔을 사용하여 모노레포 앱을 배포하는 방법을 설명합니다. Amplify는 콘솔에 지정한 앱의 루트 폴더를 사용하여 AMPLIFY_MONOREPO_APP_ROOT 환경 변수를 자동으로 설정합니다.

Amplify 콘솔을 사용하여 모노레포 앱을 배포하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 오른쪽 상단 모서리에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 공급자를 선택하고 다음을 선택합니다.
4. 리포지토리 브랜치 추가 페이지에서 다음을 수행합니다.
    a. 목록에서 리포지토리의 이름을 선택합니다.
    b. 사용할 브랜치의 이름을 선택합니다.
    c. 내 앱은 모노 리포지토리임을 선택합니다.
    d. 모노레포에 앱 경로를 입력합니다(예: **apps/app1**).
    e. 다음을 선택합니다.
5. 앱 설정 페이지에서 기본 설정을 사용하거나 앱의 빌드 설정을 사용자 지정할 수 있습니다. 환경 변수 섹션에서 Amplify가 AMPLIFY_MONOREPO_APP_ROOT를 4d단계에서 지정한 경로로 설정합니다.
6. 다음을 선택합니다.
7. 검토 페이지에서 저장 및 배포를 선택합니다.

기존 앱에 AMPLIFY_MONOREPO_APP_ROOT 환경 변수 설정

다음 지침에 따라 Amplify에 이미 배포되었거나 CloudFormation을 사용하여 만든 앱의 `AMPLIFY_MONOREPO_APP_ROOT` 환경 변수를 수동으로 설정합니다.

기존 앱에 AMPLIFY_MONOREPO_APP_ROOT 환경 변수를 설정하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 환경 변수를 설정할 앱 이름을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 환경 변수를 선택합니다.
4. 환경 변수 페이지에서 변수 관리를 선택합니다.
5. 변수 관리 섹션에서 다음을 수행합니다.
   a. 새로 추가를 선택합니다.
   b. 변수에 키 `AMPLIFY_MONOREPO_APP_ROOT`을(를) 입력합니다.
   c. 값에는 앱 경로를 입력합니다(예: **apps/app1**).
   d. 브랜치의 경우 Amplify가 기본적으로 환경 변수를 모든 브랜치에 적용합니다.
6. 저장을 선택합니다.

Turborepo 및 pnpm 모노레포 앱 구성

Turborepo와 pnpm 워크스페이스 모노레포 빌드 도구는 `.npmrc` 파일에서 구성 정보를 가져옵니다. 이러한 도구 중 하나로 만든 모노레포 앱을 배포할 때는 프로젝트 루트 디렉터리에 `.npmrc` 파일이 있어야 합니다.

`.npmrc` 파일에서 Node 패키지 설치를 위한 링커를 `hoisted`로 설정합니다. 파일에 다음 줄을 복사할 수 있습니다.

```
node-linker=hoisted
```

`.npmrc` 파일 및 설정에 대한 자세한 내용은 pnpm 설명서의 pnpm .npmrc를 참조하십시오.

Pnpm은 Amplify 기본 빌드 컨테이너에 포함되지 않습니다. pnpm 워크스페이스 및 Turborepo 앱의 경우 앱의 빌드 설정 `preBuild` 단계에서 pnpm을 설치하는 명령을 추가해야 합니다.

빌드 사양에서 발췌한 다음 예시는 pnpm 설치 명령이 포함된 `preBuild` 단계를 보여줍니다.

```
version: 1
```

```
applications:
  - frontend:
      phases:
        preBuild:
          commands:
            - npm install -g pnpm
```

빌드 이미지 사용자 지정

사용자 지정 빌드 이미지를 사용하여 Amplify 앱에 사용자 지정된 빌드 환경을 제공할 수 있습니다. Amplify 컨테이너를 사용하여 빌드 중에 설치하는 데 오랜 시간이 걸리는 특정 종속성이 있는 경우, 자체 도커 이미지를 생성하여 빌드 중에 참조할 수 있습니다. 이미지는 Amazon Elastic Container Registry Public에 호스팅될 수 있습니다.

사용자 지정 빌드 이미지가 Amplify 빌드 이미지로 작동하려면 다음 요구 사항을 충족해야 합니다.

사용자 지정 빌드 이미지 요구 사항

1. GNU C Library(glibc)를 지원하는 Linux 배포판(예: Amazon Linux)을 지원하며 x86-64 아키텍처용으로 컴파일되었습니다.
2. cURL: 사용자 지정 이미지를 시작하는 경우 빌드 러너를 컨테이너로 다운로드하므로 cURL이 있어야 합니다. 이 종속성이 누락된 경우, 빌드 러너가 출력을 생성할 수 없으므로 빌드는 출력 없이 즉시 실패합니다.
3. Git: Git 리포지토리를 복제하려면 Git를 이미지에 설치해야 합니다. 이 종속성이 누락된 경우, 리포지토리 복제 단계는 실패합니다.
4. OpenSSH: 리포지토리를 안전하게 복제하려면 OpenSSH는 빌드 중에 일시적으로 SSH 키를 설정해야 합니다. OpenSSH 패키지는 빌드 러너가 이를 수행하는 데 필요한 명령을 제공합니다.
5. Bash 및 Bourne 쉘: 이 두 가지 유틸리티는 빌드할 때 명령을 실행하는 데 사용합니다. 설치하지 않으면 시작하기도 전에 빌드가 실패할 수 있습니다.
6. Node.JS+NPM: Amazon의 빌드 러너에서는 노드를 설치하지 않습니다. 대신 이미지에 설치된 노드와 NPM을 사용합니다. 이는 NPM 패키지 또는 노드 특정 명령을 요구하는 빌드에만 필수 항목입니다. 하지만 이미 보유하고 있는 경우 Amplify 빌드 러너에서 이러한 도구를 사용하여 빌드 실행을 개선할 수 있으니 설치하는 것이 좋습니다. Amplify의 패키지 재정의 기능에서는 Hugo에 재정의를 설정할 때 NPM을 사용하여 Hugo 확장 패키지를 설치합니다.

다음과 같은 패키지는 필수는 아니나 설치하는 것이 좋습니다.

1. NVM (Node Version Manager): Node의 다른 버전을 처리해야 한다면 이 버전 관리자를 설치하는 것이 좋습니다. 재정의를 설정하면 각 빌드 전에 Amplify의 패키지 재정의 기능에서 NVM을 사용하여 Node.js 버전을 변경합니다.
2. Wget: 빌드 프로세스 도중에 Amplify에서 Wget 유틸리티를 사용하여 파일을 다운로드할 수 있습니다. 사용자 지정 이미지에 설치하는 것이 좋습니다.
3. Tar: 빌드 프로세스 도중에 Amplify에서 Tar 유틸리티를 사용하여 다운로드된 파일의 압축을 해제할 수 있습니다. 사용자 지정 이미지에 설치하는 것이 좋습니다.

앱의 사용자 지정 빌드 이미지 구성

Amplify 콘솔에서 애플리케이션의 사용자 지정 빌드 이미지를 구성하려면 다음 절차를 사용합니다.

Amazon ECR에서 호스팅되는 사용자 지정 빌드 이미지 구성하기

1. 도커 이미지를 사용하여 Amazon ECR 퍼블릭 리포지토리를 설정하려면 Amazon ECR 퍼블릭 사용 설명서의 시작하기를 참조하십시오.
2. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
3. 사용자 지정 빌드 이미지를 구성하려는 앱을 선택합니다.
4. 탐색 창에서 호스팅, 빌드 설정을 선택합니다.
5. 빌드 설정 페이지의 빌드 이미지 설정 섹션에서 편집을 선택합니다.
6. 빌드 이미지 설정 편집 페이지에서 빌드 이미지 메뉴를 펼치고 사용자 지정 이미지 빌드를 선택합니다.
7. 1단계에서 만든 Amazon ECR 퍼블릭 리포지토리 이름을 입력합니다. 여기에서 빌드 이미지가 호스팅됩니다. 예를 들어 리포지토리 이름이 ecr-examplerepo인 경우, **`public.ecr.aws/xxxxxxxx/ecr-examplerepo`**를 입력합니다.
8. 저장을 선택합니다.

빌드 이미지에서 특정 패키지 및 종속성 버전 사용

라이브 패키지 업데이트를 사용하면 Amplify 기본 빌드 이미지에 사용할 패키지 및 종속성 버전을 지정할 수 있습니다. 기본 빌드 이미지에는 사전 설치된 여러 패키지 및 종속성이 부수됩니다(예: Hugo, Amplify CLI, Yarn 등). 라이브 패키지 업데이트를 사용하면 이러한 종속성을 재정의하고 특정 버전을 지정할 수 있으며, 또는 최신 버전이 설치되어 있는지 항상 확인할 수 있습니다.

라이브 패키지 업데이트가 활성화되어 있는 경우, 빌드가 실행되기 전에 먼저 빌드 러너가 특정 종속성을 업데이트(또는 다운그레이드)합니다. 이를 통해 빌드 시간은 종속성 업데이트에 소요되는 시간에 비례하여 증가하지만 애플리케이션 빌드에 동일한 종속성 버전이 사용되도록 확인할 수 있는 장점이 있습니다.

> **Warning**
>
> Node.js 버전을 최신으로 설정하면 빌드가 실패합니다. 대신에 `18`, `21.5` 또는 `v0.1.2`처럼 정확한 Node.js 버전을 지정해야 합니다.

라이브 패키지 업데이트를 구성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 라이브 패키지 업데이트를 구성할 앱을 선택합니다.
3. 탐색 창에서 호스팅, 빌드 설정을 선택합니다.
4. 빌드 설정 페이지의 빌드 이미지 설정 섹션에서 편집을 선택합니다.
5. 빌드 이미지 설정 편집 페이지의 라이브 패키지 업데이트 목록에서 새로 추가를 선택합니다.
6. 패키지에서 재정의할 종속성을 선택합니다.
7. 버전의 경우, 기본값을 최신으로 유지하거나 종속 항목의 특정 버전을 입력하십시오. 최신이 사용될 경우, 종속성은 사용 가능한 최신 버전으로 항상 업그레이드됩니다.
8. 저장을 선택합니다.

Amplify 애플리케이션의 빌드 인스턴스 구성

Amplify Hosting은 애플리케이션의 빌드 인스턴스에 필요한 CPU, 메모리 및 디스크 공간 리소스를 제공할 수 있도록 구성 가능한 빌드 인스턴스 크기를 제공합니다. 이 기능이 출시되기 전에 Amplify는 8GiB의 메모리와 4개의 vCPUs.

Amplify는 Standard, 및 Large의 세 가지 빌드 인스턴스 유형을 지원합니다XLarge. 인스턴스 유형을 지정하지 않으면 Amplify는 기본 Standard 인스턴스를 사용합니다. Amplify 콘솔 AWS CLI, 또는 SDKs.

각 빌드 인스턴스 유형의 비용은 빌드 분당 계산됩니다. 요금에 대한 자세한 내용은 AWS Amplify 요금을 참조하세요.

다음 표에서는 각 빌드 인스턴스 유형의 컴퓨팅 사양을 설명합니다.

빌드 인스턴스 유형	vCPU	메모리	디스크 공간
Standard	4개의 vCPU	8GiB	128GB
Large	vCPUs	16GiB	128GB
XLarge	vCPUs	72GiB	256GB

주제

- 빌드 인스턴스 유형 이해
- Amplify 콘솔에서 빌드 인스턴스 유형 구성
- 대규모 인스턴스 유형을 활용하도록 애플리케이션의 힙 메모리 구성

빌드 인스턴스 유형 이해

빌드 인스턴스 유형 설정은 애플리케이션 수준에서 구성되며 애플리케이션의 모든 브랜치로 확장됩니다. 빌드 인스턴스 유형에는 다음과 같은 주요 세부 정보가 적용됩니다.

- 애플리케이션에 대해 구성하는 빌드 인스턴스 유형은 자동 생성된 브랜치 및 풀 요청 미리 보기에 자동으로 적용됩니다.
- 동시 작업 서비스 할당량은의 모든 빌드 인스턴스 유형에 적용됩니다 AWS 계정. 예를 들어 동시 작업 한도가 5인 경우의 모든 인스턴스 유형에서 최대 5개의 빌드를 실행할 수 있습니다 AWS 계정.
- 각 빌드 인스턴스 유형의 비용은 빌드 분당 계산됩니다. 빌드 인스턴스 할당 프로세스에는 빌드가 시작되기 전에 추가 오버헤드 시간이 필요할 수 있습니다. 더 큰 인스턴스, 특히 XLarge의 경우이 오버헤드 시간으로 인해 빌드가 시작되기 전에 지연 시간이 발생할 수 있습니다. 그러나 오버헤드 시간이 아닌 실제 빌드 시간에 대해서만 요금이 청구됩니다.

새 애플리케이션을 생성할 때 빌드 인스턴스 유형을 구성하거나 기존 애플리케이션에서 인스턴스 유형을 업데이트할 수 있습니다. Amplify 콘솔에서이 설정을 구성하는 방법에 대한 지침은 섹션을 참조하세요Amplify 콘솔에서 빌드 인스턴스 유형 구성. SDK를 SDKs. 자세한 내용은 Amplify APIs 참조의 CreateApp 및 UpdateApp API를 참조하세요.

사용자 지정 가능한 빌드 인스턴스 유형 기능이 출시되기 전에 생성된 기존 애플리케이션이 계정에 있는 경우 기본 Standard 인스턴스 유형을 사용합니다. 기존 애플리케이션의 빌드 인스턴스 유형을 업데이트하면 업데이트 전에 대기 중이거나 진행 중인 모든 빌드는 이전에 구성된 빌드 인스턴스 유형을

사용합니다. 예를 들어 Amplify에 배포된 `main`브랜치가 있는 기존 애플리케이션이 있고 해당 빌드 인스턴스 유형을 Standard에서 Large로 업데이트하는 경우 `main`브랜치에서 시작하는 모든 새 빌드는 Large build 인스턴스 유형을 사용합니다. 그러나 빌드 인스턴스 유형을 업데이트할 때 진행 중인 모든 빌드는 표준 인스턴스에서 계속 실행됩니다.

Amplify 콘솔에서 빌드 인스턴스 유형 구성

새 Amplify 애플리케이션을 생성할 때 다음 절차에 따라 빌드 인스턴스 유형을 구성합니다.

새 애플리케이션의 빌드 인스턴스 유형을 구성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 리포지토리 공급자를 선택하고 다음을 선택합니다.
4. 리포지토리 브랜치 추가 페이지에서 다음을 수행합니다.
   a. 최근 업데이트된 리포지토리 목록에서 연결할 리포지토리 이름을 선택합니다.
   b. 브랜치 목록에서 연결할 리포지토리 브랜치의 이름을 선택합니다.
   c. 다음을 선택합니다.
5. 앱 설정 페이지에서 고급 설정 섹션을 엽니다.
6. 빌드 인스턴스 유형의 경우 목록에서 원하는 인스턴스 유형을 선택합니다.
7. Node.js 런타임 기반 애플리케이션을 배포하는 경우 대규모 인스턴스 유형을 효과적으로 활용하도록 힙 메모리 크기를 구성합니다. 환경 변수를 설정하거나 빌드 설정을 업데이트하여 앱 설정 페이지에서이 작업을 수행할 수 있습니다. 자세한 내용은 대규모 인스턴스 유형을 활용하도록 애플리케이션의 힙 메모리 구성 단원을 참조하십시오.
   - 환경 변수 설정
     a. 고급 설정의 환경 변수 섹션에서 새로 추가를 선택합니다.
     b. 키에를 입력합니다**`NODE_OPTIONS`**.
     c. 값에 `--max-old-space-size=`*`memory_size_in_mb`*를 입력합니다. *`memory_size_in_mb`*를 원하는 힙 메모리 크기로 메가바이트 단위로 바꿉니다.
   - 빌드 설정 업데이트
     a. 빌드 설정 섹션에서 YML 파일 편집을 선택합니다.
     b. `preBuild` 단계에 다음 명령을 추가합니다. *`memory_size_in_mb`*를 원하는 힙 메모리 크기로 메가바이트 단위로 바꿉니다.

```
export NODE_OPTIONS='--max-old-space-size=memory_size_in_mb'
```

c. 저장을 선택합니다.

8. 다음을 선택합니다.
9. 검토 페이지에서 저장 및 배포를 선택합니다.

다음 절차에 따라 기존 Amplify 애플리케이션의 빌드 인스턴스 유형을 구성합니다.

기존 애플리케이션의 빌드 인스턴스 유형을 구성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 빌드 인스턴스 유형을 구성할 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택하고 빌드 설정을 선택합니다.
4. 빌드 설정 페이지의 고급 설정 섹션에서 편집을 선택합니다.
5. 설정 편집 페이지의 빌드 인스턴스 유형에서 목록에서 원하는 인스턴스 유형을 선택합니다.
6. 저장을 선택합니다. 이 변경 사항은 다음에 애플리케이션을 배포할 때 적용됩니다.
7. (선택 사항) 업데이트된 애플리케이션을 즉시 배포하려면 다음을 수행합니다.

   a. 탐색 창에서 개요를 선택합니다.

   b. 애플리케이션의 개요 페이지에서 재배포할 브랜치를 선택합니다.

   c. 배포 페이지에서 최신 배포와 같은 배포를 선택합니다. 그런 다음 이 버전 재배포를 선택합니다. 새 배포가 시작됩니다.

   d. 배포가 완료되면 애플리케이션의 빌드 설정에 브랜치가 업데이트된 빌드 인스턴스 유형을 사용하고 있는 것으로 표시됩니다.

대규모 인스턴스 유형을 활용하도록 애플리케이션의 힙 메모리 구성

메모리 집약적인 애플리케이션을 구축하는 경우이 섹션을 사용하여 대규모 인스턴스 유형을 활용하도록 애플리케이션을 구성하는 방법을 이해합니다. 프로그래밍 언어 및 프레임워크는 런타임 중에 힙 메모리라고도 하는 동적 메모리를 할당하여 애플리케이션 메모리 요구 사항을 관리하는 데 의존하는 경우가 많습니다. 힙 메모리는 런타임 환경에서 요청하고 호스트 운영 체제에서 할당합니다. 기본적으로 런타임 환경은 애플리케이션에 사용할 수 있는 최대 힙 크기 제한을 적용합니다. 즉, 호스트 운영 체제 또는 컨테이너에 사용 가능한 메모리 양이 더 많더라도 힙 크기를 초과하는 애플리케이션에는 추가 메모리를 사용할 수 없습니다.

예를 들어 JavaScript Node.js v8 런타임 환경은 호스트 메모리 크기를 비롯한 여러 요인에 따라 기본 힙 크기 제한을 적용합니다. 따라서 Standard 및 Large 빌드 인스턴스의 기본 Node.js 힙 크기는 2,096MB이고 XLarge 인스턴스의 기본 힙 크기는 4,144MB입니다. 따라서 Amplify 빌드 인스턴스 유형에서 기본 Node.js 힙 크기를 사용하여 메모리 요구 사항이 6,000MB인 애플리케이션을 빌드하면 out-of-memory 오류로 인해 빌드가 실패합니다.

Node.js 기본 힙 크기 메모리 제한을 해결하려면 다음 중 하나를 수행할 수 있습니다.

- Amplify 애플리케이션의 `NODE_OPTIONS` 환경 변수를 값으로 설정합니다`--max-old-space-size=`*`memory_size_in_mb`*. 의 경우 원하는 힙 메모리 크기를 메가바이트 단위로 `memory_size_in_mb`지정합니다.

  지침은 환경 변수 설정 섹션을 참조하세요.
- Amplify 애플리케이션 빌드 사양의 `preBuild` 단계에 다음 명령을 추가합니다.

  ```
  export NODE_OPTIONS='--max-old-space-size=memory_size_in_mb'
  ```

  Amplify 콘솔 또는 프로젝트 리포지토리의 애플리케이션 `amplify.yml` 파일에서 빌드 사양을 업데이트할 수 있습니다. 지침은 Amplify 애플리케이션의 빌드 설정 구성 섹션을 참조하세요.

  다음 Amplify 빌드 사양 예제에서는 React 프런트엔드 애플리케이션을 빌드하기 위해 Node.js 힙 메모리 크기를 7,000MB로 설정합니다.

  ```
  version: 1
  frontend:
    phases:
      preBuild:
        commands:
          # Set the heap size to 7000 MB
          - export NODE_OPTIONS='--max-old-space-size=7000'
          # To check the heap size memory limit in MB
          - node -e "console.log('Total available heap size (MB):',
   v8.getHeapStatistics().heap_size_limit / 1024 / 1024)"
          - npm ci --cache .npm --prefer-offline
      build:
        commands:
          - npm run build
    artifacts:
      baseDirectory: build
      files:
        - '**/*'
  ```

```
cache:
  paths:
    - .npm/**/*
```

대규모 인스턴스 유형을 효과적으로 활용하려면 충분한 힙 메모리 크기를 구성해야 합니다. 메모리 집약적인 애플리케이션에 대해 작은 힙 크기를 구성하면 빌드에 실패할 수 있습니다. 애플리케이션 런타임이 예기치 않게 충돌할 수 있으므로 애플리케이션의 빌드 로그는 out-of-memory 오류를 직접 나타내지 않을 수 있습니다. 호스트 메모리만큼 힙 크기를 구성하면 호스트 운영 체제가 다른 프로세스를 교체하거나 종료하여 빌드 프로세스가 중단될 수 있습니다. 참고로 Node.js는 약 2,000MB의 메모리가 있는 시스템에서 최대 힙 크기를 1,536MB로 설정하여 일부 메모리를 다른 용도로 남겨둘 것을 권장합니다.

최적의 힙 크기는 애플리케이션의 요구 사항과 리소스 사용량에 따라 달라집니다. out-of-memory 오류가 발생하는 경우 적당한 힙 크기로 시작한 다음 필요에 따라 점진적으로 늘리세요. 지침에 따라 인스턴스 유형의 경우 6000MB, Standard 인스턴스 유형의 경우 12000MB, Large 인스턴스 유형의 경우 60000MB로 시작하는 것이 좋습니다XLarge.

빌드를 시작하기 위한 수신 웹후크 생성

Git 리포지토리에 코드를 커밋하지 않고 빌드를 시작할 수 있도록 Amplify 콘솔에 수신 웹후크를 설정합니다. 웹후크는 콘텐츠 변경에 관한 빌드를 트리거하거나 Zapier와 같은 서비스를 사용하여 일상적인 빌드를 트리거하기 위해 헤드리스 CMS 도구(예: Contentful 또는 GraphCMS)와 함께 사용될 수 있습니다.

수신 웹후크를 만들려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 웹후크를 생성할 앱을 선택합니다.
3. 탐색 창에서 호스팅, 빌드 설정을 선택합니다.
4. 빌드 설정 페이지에서 수신 웹후크 섹션으로 스크롤하여 웹후크 생성을 선택합니다.
5. 웹후크 생성 대화 상자에서 다음 작업을 수행하십시오.
   a. 웹후크 이름에는 웹후크 이름을 입력합니다.
   b. 브랜치를 빌드하려면 들어오는 웹후크 요청에 빌드할 브랜치를 선택합니다.
   c. 웹후크 생성을 선택합니다.
6. 수신 웹후크 섹션에서 다음 중 하나를 수행하십시오.

- 웹후크 URL을 복사하여 헤드리스 CMS 도구 또는 기타 서비스에 제공하여 빌드를 시작합니다.
- 터미널 창에서 curl 명령을 실행하여 새 빌드를 시작합니다.

빌드에 대한 이메일 알림 설정

빌드 성공 또는 실패 시 이해관계자 또는 팀원에게 알리도록 AWS Amplify 앱에 대한 이메일 알림을 설정할 수 있습니다. Amplify Hosting은 계정에서 Amazon Simple Notification Service(SNS)항목을 생성하고 이를 사용하여 이메일 알림을 구성합니다. Amplify 앱의 모든 분기 또는 특정 분기에 적용되도록 알림을 구성할 수 있습니다.

이메일 알림 설정

다음 절차를 사용하여 Amplify 앱의 모든 분기 또는 특정 분기에 대해 이메일 알림을 설정합니다.

Amplify 앱의 이메일 알림을 설정하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 이메일 알림을 설정하려는 앱을 선택합니다.
3. 탐색 창에서 호스팅, 빌드 알림를 선택합니다. 빌드 알림 페이지에서 알림 관리를 선택합니다.
4. 알림 관리 페이지에서 새로 추가를 선택합니다.
5. 다음 중 하나를 수행합니다.
   - 단일 브랜치에 알림을 보내려면 이메일에 알림을 보낼 이메일 주소를 입력합니다. 브랜치의 경우, 알림을 보낼 브랜치 이름을 선택합니다.
   - 연결된 모든 브랜치에 알림을 보내려면 이메일에 알림을 보낼 이메일 주소를 입력합니다. 브랜치의 경우, 모든 브랜치를 선택합니다.
6. 저장을 선택합니다.

사용자 지정 도메인 연결

Amplify Hosting으로 배포한 앱을 사용자 지정 도메인에 연결할 수 있습니다. Amplify를 사용하여 웹 앱을 배포하는 경우 Amplify는 `https://branch-name.d1m7bkiki6tdw1.amplifyapp.com`과 같은 URL에서 기본 `amplifyapp.com` 도메인을 호스팅합니다. 앱을 사용자 지정 도메인에 연결하면 사용자는 앱이 `https://www.example.com`과 같은 사용자 지정 URL에서 호스팅되는 것을 볼 수 있습니다.

공인 도메인 등록 기관(예: Amazon Route 53 또는 GoDaddy)을 통해 사용자 지정 도메인을 구입할 수 있습니다. Route 53는 아마존의 DNS (도메인 이름 시스템) 웹 서비스입니다. Route 53의 사용에 대한 자세한 내용은 Amazon Route 53이란을 참조하십시오. 공인 도메인 등록 기관의 목록은 ICANN 웹 사이트의 Accredited Registrar Directory를 참조하세요.

사용자 지정 도메인을 설정할 때 Amplify가 제공하는 기본 관리형 인증서를 사용하거나 자체 사용자 지정 인증서를 사용할 수 있습니다. 도메인에 사용 중인 인증서는 언제든지 변경할 수 있습니다. 인증서 관리에 대한 자세한 내용은 SSL/TLS 인증서 사용 섹션을 참조하세요.

사용자 지정 도메인 설정을 진행하기 전에 다음 사전 조건을 충족하는지 확인합니다.

- 등록된 도메인 이름을 소유하고 있습니다.
- 에서 발급하거나 로 가져온 인증서가 있습니다 AWS Certificate Manager.
- 앱을 Amplify Hosting에 배포했습니다.

  이 단계를 완료하는 방법에 대한 자세한 내용은 Amplify Hosting에 앱 배포 시작하기 섹션을 참조하세요.
- 도메인 및 DNS 용어에 대한 기본 지식이 있습니다.

  도메인 및 DNS에 대한 자세한 내용은 DNS 용어 및 개념 이해을 참조하십시오.

주제

- 도메인의 SSL/TLS 인증서 업데이트
- 하위 도메인 관리
- 와일드카드 하위 도메인 설정
- Amazon Route 53 사용자 지정 도메인을 위한 자동 하위 도메인 설정
- 사용자 지정 도메인 문제 해결

DNS 용어 및 개념 이해

도메인 이름 시스템(DNS)과 관련된 용어 및 개념에 익숙하지 않은 경우, 다음 항목을 참조하여 사용자 지정 도메인을 추가하는 절차를 이해하는 데 도움이 될 수 있습니다.

DNS 용어

다음은 DNS에서 흔히 사용되는 용어 목록입니다. 사용자 지정 도메인을 추가하는 절차를 이해하는 데 도움이 될 수 있습니다.

CNAME

정식 레코드 이름(CNAME)은 웹페이지 세트에 대해 도메인을 숨기고 다른 곳에 위치해 있는 것처럼 표시할 수 있게 해주는 DNS 레코드의 유형입니다. CNAME은 정규화된 도메인 이름(FQDN)에 대한 하위 도메인을 가리킵니다. 예를 들어, 새 CNAME 레코드를 생성하여 하위 도메인 www.example.com(여기서 www는 하위 도메인임)을 Amplify 콘솔에서 앱에 할당된 FQDN 도메인 branch-name.d1m7bkiki6tdw1.cloudfront.net에 매핑할 수 있습니다.

ANAME

ANAME 레코드는 CNAME 레코드와 같지만 루트 수준에 해당합니다. ANAME는 FQDN에 대한 도메인의 루트를 가리킵니다. 해당 FQDN은 IP 주소를 가리킵니다.

이름 서버

이름 서버는 도메인 이름의 다양한 서비스의 위치에 관한 쿼리를 전문적으로 처리하는 인터넷상의 서버입니다. Amazon Route 53에서 도메인을 설정하면 도메인에 이름 서버 목록이 이미 할당되어 있습니다.

NS 레코드

NS 레코드는 도메인 세부 정보를 조회하는 네임 서버를 가리킵니다.

DNS 확인

도메인 이름 시스템(DNS)은 사람이 읽을 수 있는 도메인 이름을 컴퓨터 친화적인 IP 주소로 변환하는 전화번호부와 같습니다. **`https://google.com`**을(를) 브라우저에 입력하면 DNS 공급자에서 웹 사이트를 호스팅하는 서버의 IP 주소를 찾기 위한 조회 작업이 수행됩니다.

DNS 공급자는 도메인 레코드와 해당 IP 주소를 포함합니다. 가장 일반적으로 사용되는 DNS 레코드는 CNAME, ANAME, NS 레코드입니다.

Amplify는 CNAME 레코드를 사용하여 사용자가 자체 사용자 지정 도메인을 소유하고 있는지 확인합니다. Route 53을 사용하여 도메인을 호스팅하는 경우, 자동으로 사용자를 대신하여 확인이 이루어집니다. 하지만 GoDaddy과 같은 타사 공급자를 통해 도메인을 호스팅하는 경우, 도메인의 DNS 설정을 수동으로 업데이트하고 Amplify에서 제공하는 새 CNAME 레코드를 추가해야 합니다.

사용자 지정 도메인 활성화 프로세스

Amplify 콘솔에서 Amplify 앱을 사용자 지정 도메인에 연결하는 경우, Amplify가 몇 가지 단계를 완료해야 사용자 지정 도메인을 사용하여 앱을 볼 수 있습니다. 다음 목록은 도메인 설정 및 활성화 프로세스의 각 단계를 자세히 설명합니다.

SSL/TLS 생성

관리형 인증서를 사용하는 경우는 보안 사용자 지정 도메인을 설정하기 위해 SSL/TLS 인증서를 AWS Amplify 발급합니다.

SSL/TLS 구성 및 확인

Amplify는 관리형 인증서를 발급하기 이전에 사용자가 도메인 소유자인지 확인합니다. Amazon Route 53에서 관리하는 도메인의 경우, Amplify에서 DNS 확인 레코드를 자동으로 업데이트합니다. Route 53 외부에서 관리하는 도메인의 경우, Amplify 콘솔에서 제공된 DNS 확인 레코드를 도메인의 타사 DNS 공급자에게 수동으로 추가해야 합니다.

사용자 지정 인증서를 사용하는 경우 사용자에게 도메인 소유권을 검증할 책임이 있습니다.

도메인 활성화

도메인이 성공적으로 확인되었습니다. Route 53 외부에서 관리하는 도메인의 경우, Amplify 콘솔에서 제공되는 CNAME 레코드를 도메인의 타사 DNS 공급자에게 수동으로 추가해야 합니다.

SSL/TLS 인증서 사용

SSL/TLS 인증서는 웹 브라우저가 보안 SSL/TLS 프로토콜을 사용하여 웹 사이트에 대한 암호화된 네트워크 연결을 식별하고 설정할 수 있도록 허용하는 디지털 문서입니다. 사용자 지정 도메인을 설정할 때 Amplify가 제공하는 기본 관리형 인증서를 사용하거나 자체 사용자 지정 인증서를 사용할 수 있습니다.

관리형 인증서를 사용하는 경우 Amplify는 앱에 연결된 모든 도메인에 대해 SSL/TLS 인증서를 발급하여 모든 트래픽을 HTTPS/2를 통해 보호합니다. AWS Certificate Manager (ACM)에서 생성한 기본 인증서는 13개월 동안 유효하며 앱이 Amplify로 호스팅되는 한 자동으로 갱신됩니다.

> **Warning**
>
> 도메인 공급자의 DNS 설정에서 CNAME 확인 레코드가 수정 또는 삭제된 경우 Amplify가 인증서를 갱신할 수는 없습니다. Amplify 콘솔에서 도메인을 삭제하고 다시 추가해야 합니다.

사용자 지정 인증서를 사용하려면 먼저 선택한 타사 인증 기관으로부터 인증서를 발급받아야 합니다. Amplify Hosting은 RSA(Rivest-Shamir-Adleman) 및 ECDSA(Elliptic Curve Digital Signature Algorithm)의 두 가지 유형의 인증서를 지원합니다. 각 인증서 유형은 다음 요구 사항을 준수해야 합니다.

RSA 인증서

- Amplify Hosting은 1024비트, 2048비트, 3072비트 및 4096비트 RSA 키를 지원합니다.
- AWS Certificate Manager (ACM)는 최대 2048비트 키로 RSA 인증서를 발급합니다.
- 3072비트 또는 4096비트 RSA 인증서를 사용하려면 인증서를 외부에서 발급받아 ACM으로 가져옵니다. 그러면 Amplify Hosting과 함께 사용할 수 있습니다.

ECDSA 인증서

- Amplify Hosting은 256비트 키를 지원합니다.
- Prime256v1 타원 곡선을 사용하여 Amplify Hosting에 대한 ECDSA 인증서를 가져옵니다.

인증서를 받은 후 로 가져옵니다 AWS Certificate Manager. ACM은 및 내부 연결 리소스와 함께 사용할 퍼블릭 및 프라이빗 SSL/TLS 인증서를 쉽게 프로비저닝, 관리 AWS 서비스 및 배포할 수 있는 서비스입니다. 미국 동부(버지니아 북부)(us-east-1) 리전에서 인증서를 요청하거나 가져와야 합니다

사용자 지정 인증서가 추가하려는 모든 하위 도메인을 포함하는지 확인합니다. 도메인 이름 시작 부분에 와일드카드를 사용하여 여러 하위 도메인을 포함할 수 있습니다. 예를 들어 도메인이 `example.com`인 경우 와일드카드 도메인 `*.example.com`을 포함할 수 있습니다. 여기에는 `product.example.com` 및 `api.example.com`과 같은 하위 도메인이 포함됩니다.

ACM에서 사용자 지정 인증서를 사용할 수 있게 되면 도메인 설정 프로세스 중에 해당 인증서를 선택할 수 있습니다. 인증서를 AWS Certificate Manager로 가져오는 방법에 대한 지침은 AWS Certificate Manager 사용 설명서의 AWS Certificate Manager로 인증서 가져오기를 참조하세요.

ACM에서 사용자 지정 인증서를 갱신하거나 다시 가져오면 Amplify는 사용자 지정 도메인과 연결된 인증서 데이터를 새로 고칩니다. 가져온 인증서의 경우 ACM은 갱신을 자동으로 관리하지 않습니다. 사용자 지정 인증서를 갱신하고 다시 가져올 책임은 사용자에게 있습니다.

도메인에 사용 중인 인증서는 언제든지 변경할 수 있습니다. 예를 들어 기본 관리형 인증서에서 사용자 지정 인증서로 전환하거나 사용자 지정 인증서에서 관리형 인증서로 변경할 수 있습니다. 또한 사용 중인 사용자 지정 인증서를 다른 사용자 지정 인증서로 변경할 수 있습니다. 인증서 업데이트에 대한 지침은 도메인의 SSL/TLS 인증서 업데이트를 참조하세요.

Amazon Route 53에서 관리하는 사용자 지정 도메인 추가

Amazon Route 53는 가용성 및 확장성이 뛰어난 DNS 서비스입니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 Amazon Route 53를 참조하세요. Route 53 도메인이 이미 있는 경우 다음 지침을 사용하여 사용자 지정 도메인을 Amplify 앱에 연결합니다.

Route 53에서 관리하는 사용자 지정 도메인을 추가하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 사용자 지정 도메인에 연결할 앱을 선택합니다.
3. 탐색 창에서 호스팅, 사용자 도메인을 선택합니다.
4. 사용자 지정 도메인 페이지에서 도메인 추가를 선택합니다.
5. 루트 도메인의 이름을 입력합니다. 예를 들어, 도메인 이름이 https://example.com인 경우, **`example.com`**을(를) 입력합니다.

   입력을 시작하면 Route 53에서 이미 관리하고 있는 모든 루트 도메인이 목록에 나타납니다. 목록에서 사용할 도메인을 선택할 수 있습니다. 아직 도메인을 소유하고 있지 않고 사용 가능한 경우, Amazon Route 53에서 도메인을 구매할 수 있습니다.
6. 도메인 이름을 입력한 후 도메인 구성을 선택합니다.

7. 기본적으로 Amplify는 도메인에 대해 두 개의 하위 도메인 항목을 자동으로 생성합니다. 예를 들어 도메인 이름이 example.com인 경우, 루트 도메인에서 www 하위 도메인으로 리디렉션이 설정된 하위 도메인 https://www.example.com 및 https://example.com이 표시됩니다.

   (선택 사항) 하위 도메인만을 추가하려는 경우, 기본 구성을 수정할 수 있습니다. 하위 도메인 연결을 참조하십시오. 기본 구성을 변경하려면 탐색 창에서 다시 쓰기 및 리디렉션을 선택한 다음 도메인을 구성합니다.

8. 사용할 SSL/TLS 인증서를 선택합니다. Amplify에서 프로비저닝하는 기본 관리형 인증서 또는 가져온 사용자 지정 타사 인증서를 사용할 수 있습니다 AWS Certificate Manager.

   - 기본 Amplify 관리형 인증서 사용
     - Amplify 관리형 인증서를 선택합니다.
   - 사용자 지정 타사 인증서 사용
     a. 사용자 지정 SSL 인증서를 선택합니다.
     b. 목록에서 사용할 인증서를 선택합니다.

9. 도메인 추가를 선택합니다.

> **Note**
>
> DNS 전파와 인증서 발급에 최대 24시간이 소요될 수 있습니다. 발생한 오류를 해결하는 데 도움이 필요하면 사용자 지정 도메인 문제 해결 단원을 참조하십시오.

타사 DNS 공급자가 관리하는 사용자 지정 도메인 추가

Amazon Route 53을 사용하여 도메인을 관리하지 않는 경우, Amplify로 배포한 앱에 타사 DNS 공급자가 관리하는 사용자 지정 도메인을 추가할 수 있습니다.

GoDaddy를 사용하는 경우 이 공급자에 대한 지침은 the section called “GoDaddy에서 관리하는 도메인의 DNS 레코드 업데이트” 섹션을 참조하세요.

타사 DNS 공급자가 관리하는 사용자 지정 도메인 추가

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 사용자 지정 도메인을 추가할 앱을 선택합니다.
3. 탐색 창에서 호스팅, 사용자 도메인을 선택합니다.

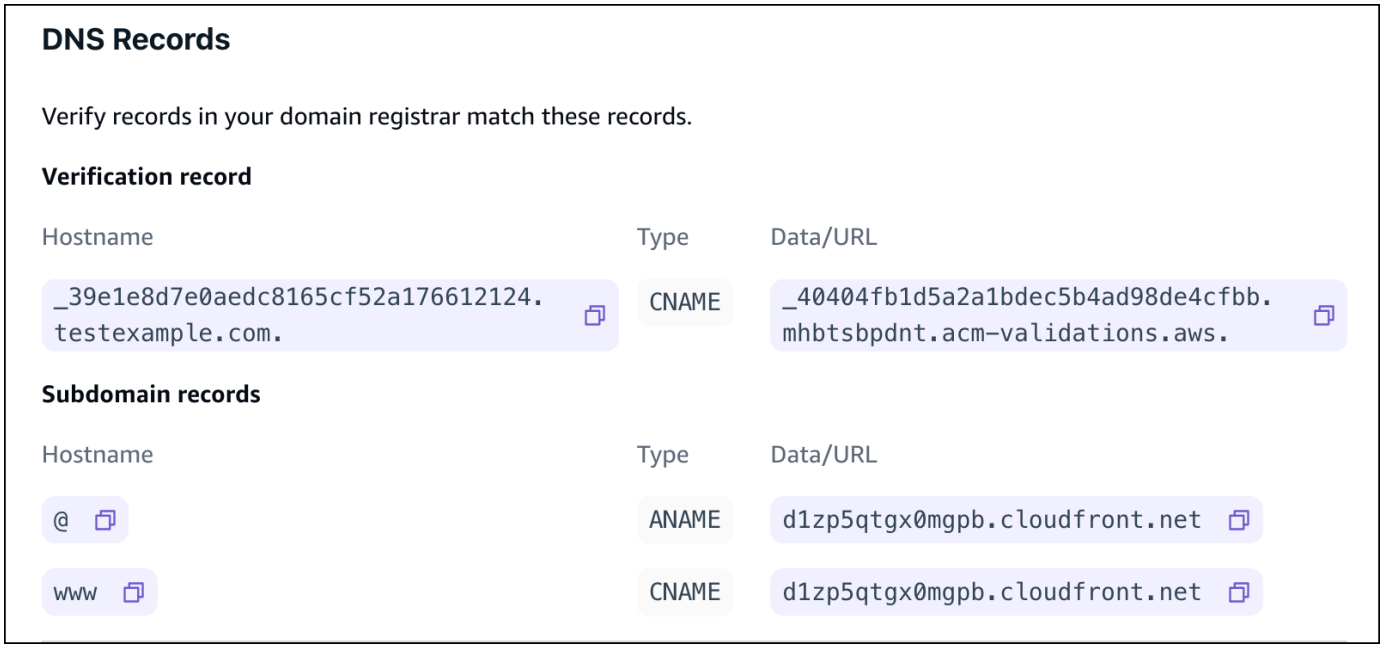
4. 사용자 지정 도메인 페이지에서 도메인 추가를 선택합니다.
5. 루트 도메인의 이름을 입력합니다. 예를 들어, 도메인 이름이 https://example.com인 경우, **example.com**을(를) 입력합니다.
6. Amplify는 Route 53 도메인을 사용하지 않음을 감지하고 Route 53에서 호스팅 영역을 생성할 수 있는 옵션을 제공합니다.
   - Route 53에서 호스팅 영역을 생성하려면
     a. Route 53에서 호스팅 영역 생성을 선택합니다.
     b. 도메인 구성을 선택합니다.
     c. 콘솔에 호스팅 영역 이름 서버가 표시됩니다. DNS 공급자의 웹 사이트로 이동하여 이름 서버를 DNS 설정에 추가합니다.
     d. 위의 이름 서버를 내 도메인 레지스트리에 추가했음을 선택합니다.
     e. 7단계로 진행합니다.
   - 수동 구성을 계속하려면
     a. 수동 구성을 선택합니다.
     b. 도메인 구성을 선택합니다.
     c. 7단계로 진행합니다.
7. 기본적으로 Amplify는 도메인에 대해 두 개의 하위 도메인 항목을 자동으로 생성합니다. 예를 들어 도메인 이름이 example.com인 경우, 루트 도메인에서 www 하위 도메인으로 리디렉션이 설정된 하위 도메인 https://www.example.com 및 https://example.com이 표시됩니다.

   (선택 사항) 하위 도메인만을 추가하려는 경우, 기본 구성을 수정할 수 있습니다. 하위 도메인 연결을 참조하십시오. 기본 구성을 변경하려면 탐색 창에서 다시 쓰기 및 리디렉션을 선택한 다음 도메인을 구성합니다.
8. 사용할 SSL/TLS 인증서를 선택합니다. Amplify에서 프로비저닝하는 기본 관리형 인증서 또는 가져온 사용자 지정 타사 인증서를 사용할 수 있습니다 AWS Certificate Manager.
   - 기본 Amplify 관리형 인증서 사용
     - Amplify 관리형 인증서를 선택합니다.
   - 사용자 지정 타사 인증서 사용
     a. 사용자 지정 SSL 인증서를 선택합니다.
     b. 목록에서 사용할 인증서를 선택합니다.
9. 도메인 추가를 선택합니다.

10. 6단계에서 Route 53에서 호스팅 영역 생성을 선택한 경우 15단계로 진행합니다.

    수동 구성을 선택한 경우 6단계에서 DNS 레코드를 타사 도메인 공급자로 업데이트해야 합니다.

    작업 메뉴에서 DNS 레코드 보기를 선택합니다. 다음 스크린샷은 콘솔에 표시된 DNS 레코드를 보여줍니다.



11. 다음 중 하나를 수행합니다.

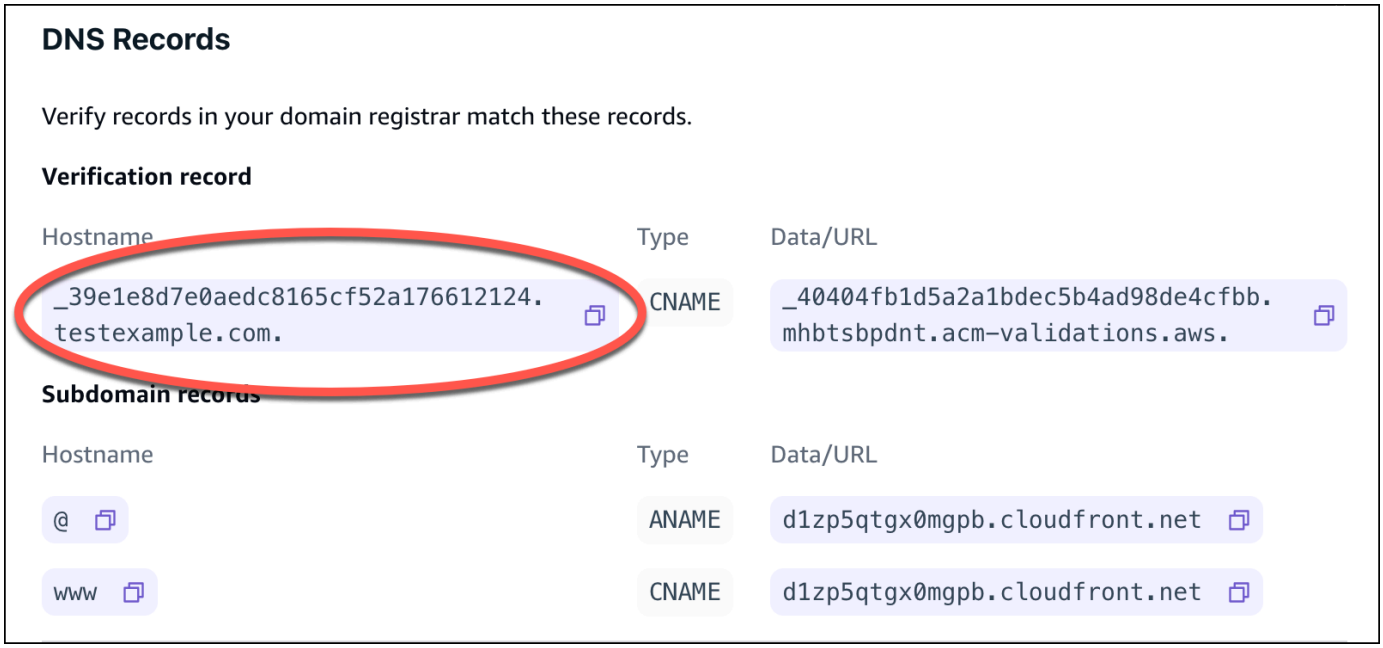
    - GoDaddy를 사용하는 경우, GoDaddy에서 관리하는 도메인의 DNS 레코드 업데이트로 이동하십시오.
    - 다른 타사 DNS 공급자를 사용하는 경우, 이 절차의 다음 단계로 이동하십시오.

12. DNS 공급자의 웹사이트로 이동하여 계정에 로그인한 다음 도메인의 DNS 관리 설정을 찾으십시오. 두 개의 CNAME 레코드를 구성해야 합니다.

13. 하위 도메인이 AWS 검증 서버를 가리키도록 첫 번째 CNAME 레코드를 구성합니다.

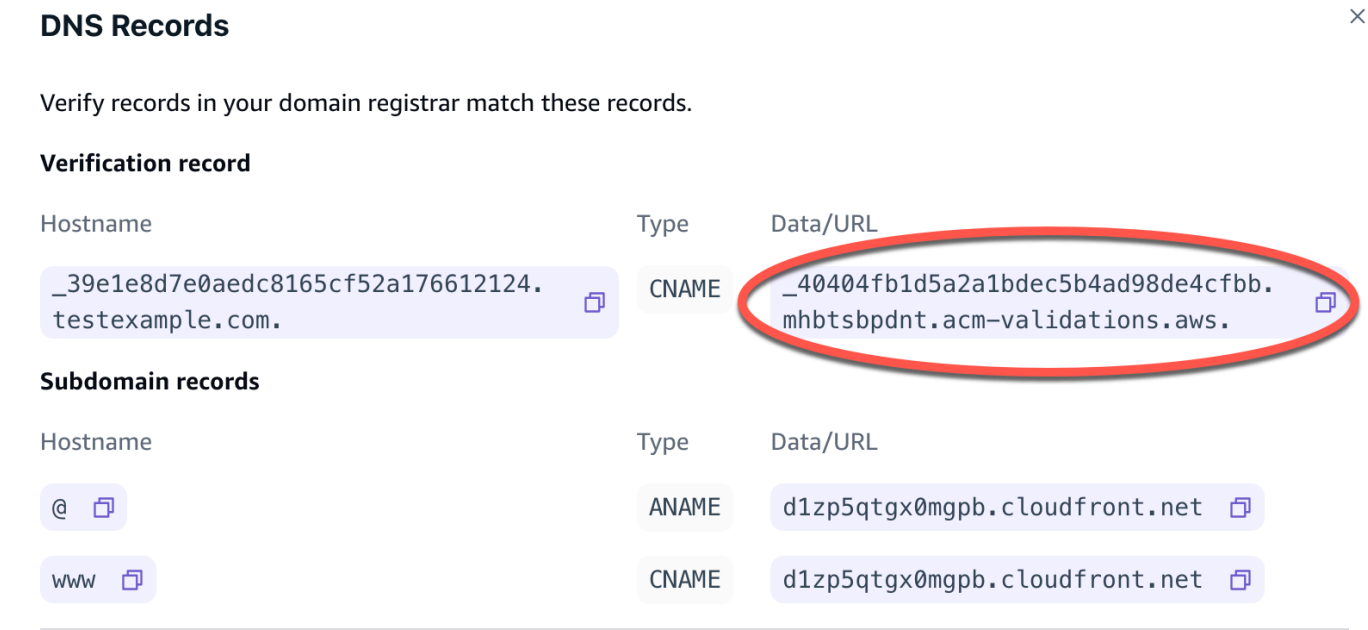
    Amplify 콘솔에 _c3e2d7eaf1e656b73f46cd6980fdc0e.example.com 같은 하위 도메인의 소유권을 확인하기 위한 DNS 레코드가 표시되는 경우 CNAME 레코드 하위 도메인 이름에 **`_c3e2d7eaf1e656b73f46cd6980fdc0e`**만 입력합니다.

    다음 스크린샷은 사용할 확인 레코드의 위치를 보여줍니다.



Amplify 콘솔에 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws와 같은 ACM 검증 서버 레코드가 표시되면 CNAME 레코드 값에 **_cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws**를 입력합니다.

다음 스크린샷은 사용할 ACM 확인 레코드의 위치를 보여줍니다.



Amplify는 이 정보를 사용하여 도메인 소유권을 확인하고 도메인의 SSL/TLS 인증서를 생성합니다. Amplify이 도메인의 소유권을 확인하면 모든 트래픽이 HTTPS/2를 통해 제공됩니다.

> **Note**
>
> AWS Certificate Manager (ACM)에서 생성한 기본 Amplify 인증서는 13개월 동안 유효하며 앱이 Amplify로 호스팅되는 한 자동으로 갱신됩니다. CNAME 확인 레코드가 수정 또는 삭제된 경우, Amplify는 인증서를 갱신할 수 없습니다. Amplify 콘솔에서 도메인을 삭제하고 다시 추가해야 합니다.

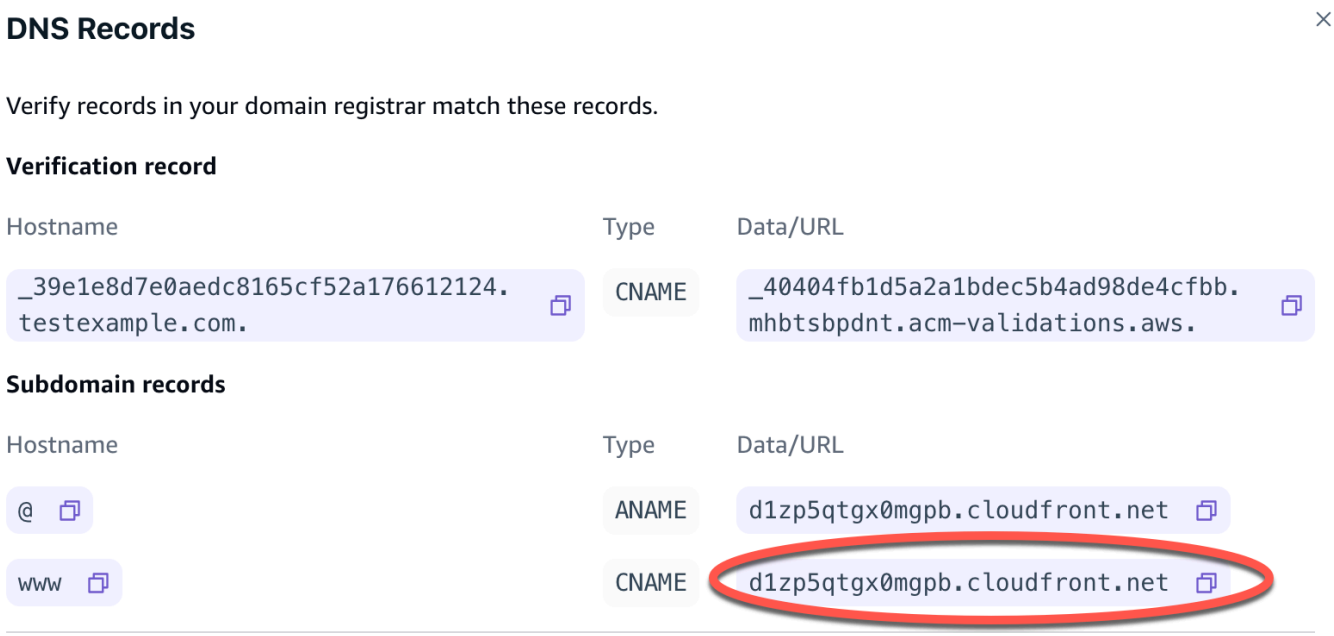
> **Important**
>
> Amplify 콘솔에서 사용자 지정 도메인을 추가한 후 바로 이 단계를 수행하는 것이 중요합니다. AWS Certificate Manager (ACM)는 소유권 확인을 즉시 시작합니다. 시간이 지나면 검사 빈도가 줄어듭니다. 앱을 만든 후 몇 시간 후에 CNAME 레코드를 추가하거나 업데이트하면 앱이 확인 보류 중 상태에서 멈출 수 있습니다.

14. 하위 도메인이 Amplify 도메인을 가리키도록 두 번째 CNAME 레코드를 구성합니다. 예를 들어, 하위 도메인이 www.example.com인 경우 하위 도메인 이름에 www를 입력합니다.

    Amplify 콘솔에 앱의 도메인이 d111111abcdef8.cloudfront.net로 표시되면 Amplify 도메인에 **`d111111abcdef8.cloudfront.net`**를 입력합니다.

    프로덕션 트래픽이 있는 경우, 도메인 상태가 Amplify 콘솔에서 AVAILABLE을 표시한 후 CNAME 레코드를 업데이트하는 것이 좋습니다.

    다음 스크린샷은 사용할 도메인 이름 레코드의 위치를 보여줍니다.



15. 앱의 루트 도메인(예: https://example.com)을 가리키도록 ANAME/ALIAS 레코드를 구성합니다. ANAME 레코드는 도메인의 루트가 호스트 이름을 가리킬 수 있습니다. 프로덕션 트래픽이 있는 경우, 도메인 상태가 콘솔에서 AVAILABLE을 표시한 후 ANAME 레코드를 업데이트하는 것이 좋습니다.* ANAME/ALIAS 지원 기능이 없는 DNS 공급자의 경우, DNS를 Route 53으로 마이그레이션하는 것이 좋습니다. 자세한 내용은 Amazon Route 53을 DNS 서비스로 구성을 참조하십시오.

> **Note**
>
> 타사 도메인에 대한 도메인 소유권 및 DNS 전파 확인에는 최대 48시간이 소요될 수 있습니다. 발생하는 오류를 해결하는 데 도움이 필요하면 사용자 지정 도메인 문제 해결을 참조하십시오.

GoDaddy에서 관리하는 도메인의 DNS 레코드 업데이트

GoDaddy가 DNS 공급자인 경우 다음 지침을 사용하여 GoDaddy UI에서 DNS 레코드를 업데이트하여 Amplify 앱을 GoDaddy 도메인에 연결하는 작업을 완료합니다.

GoDaddy에서 관리하는 사용자 지정 도메인을 추가하려면

1. GoDaddy를 사용하여 DNS 레코드를 업데이트하려면 먼저 절차 the section called “타사 DNS 공급자가 관리하는 사용자 지정 도메인 추가”의 1~9단계를 완료합니다.
2. GoDaddy 계정에 로그인합니다.
3. 도메인 목록에서 추가할 도메인을 찾고 DNS 관리를 선택합니다.

4. GoDaddy는 DNS 페이지의 DNS 레코드 섹션에 도메인의 레코드 목록을 표시합니다. 새 CNAME 레코드 두 개를 추가해야 합니다.
5. 첫 번째 CNAME 레코드를 생성하여 하위 도메인이 Amplify 도메인을 가리키도록 합니다.

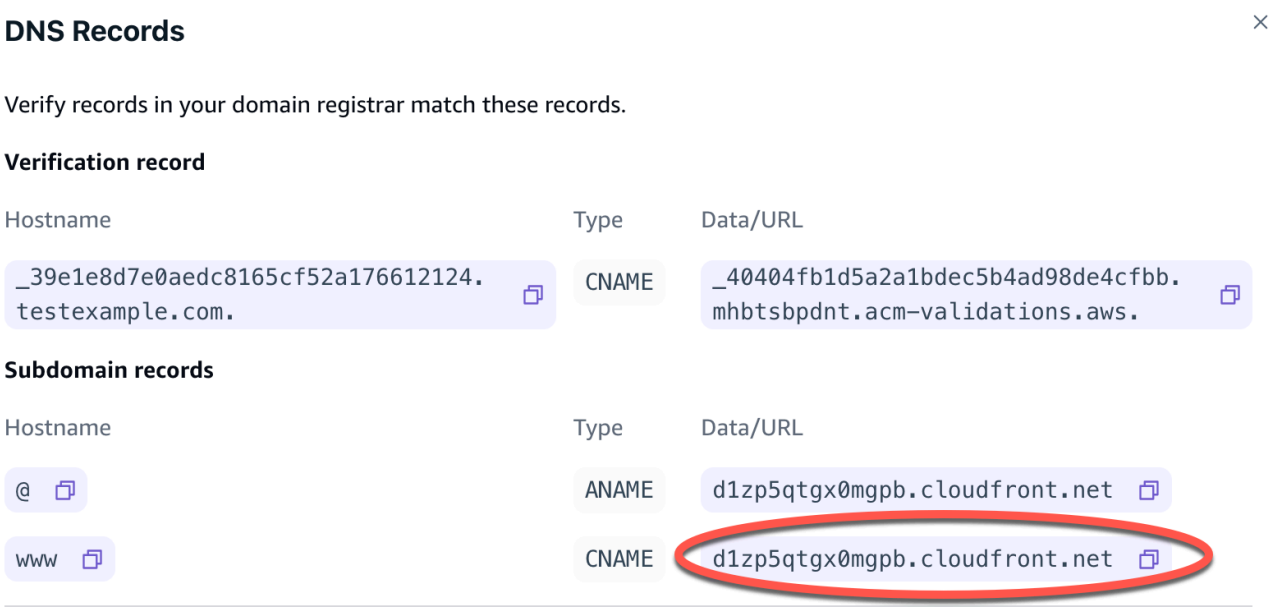
   a. DNS 레코드 섹션에서 새 레코드 추가를 선택합니다.

   b. 유형에서 CNAME을 선택합니다.

   c. 이름에는 하위 도메인만 입력합니다. 예를 들어, 하위 도메인이 www.example.com인 경우, 이름에 www를 입력합니다.

   d. 값의 경우, Amplify 콘솔에서 DNS 레코드를 확인한 다음 값을 입력합니다. Amplify 콘솔에 앱의 도메인이 d111111abcdef8.cloudfront.net로 표시되면 값에 **`d111111abcdef8.cloudfront.net`**를 입력합니다.

   다음 스크린샷은 사용할 도메인 이름 레코드의 위치를 보여줍니다.



   e. 저장을 선택합니다.

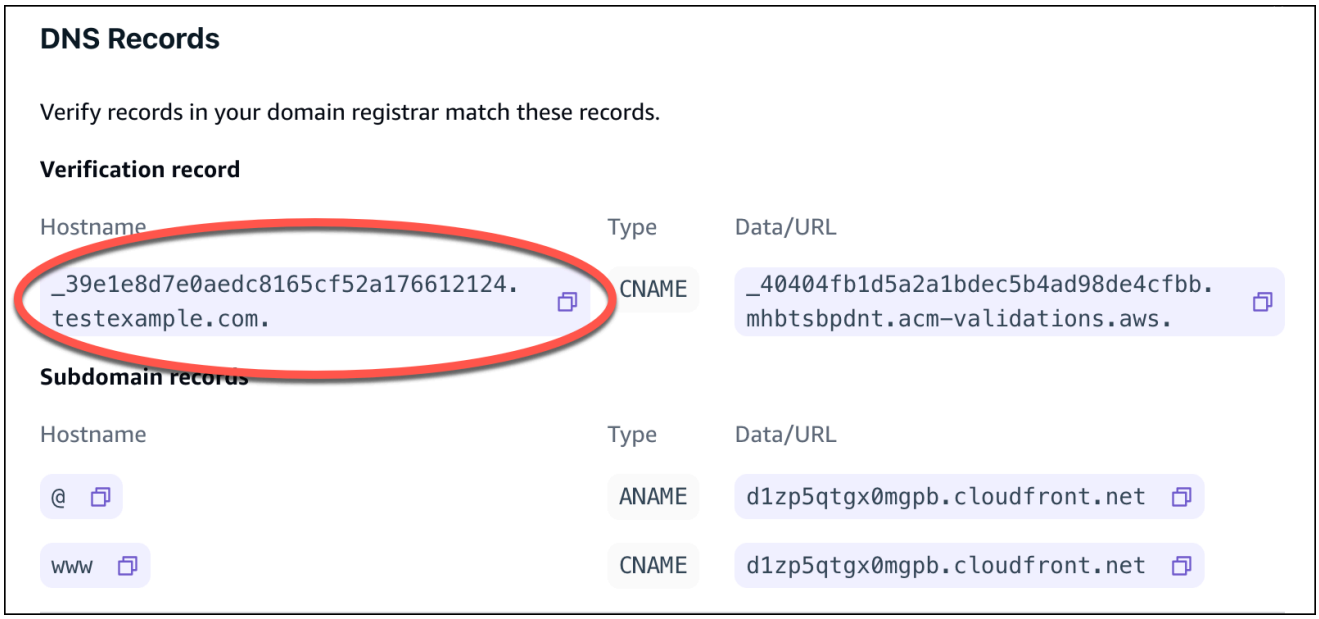
6. AWS Certificate Manager (ACM) 검증 서버를 가리키는 두 번째 CNAME 레코드를 생성합니다. 확인된 단일 ACM이 도메인의 SSL/TLS 인증서를 생성합니다.

   a. 유형에서 CNAME을 선택합니다.

   b. 이름에 하위 도메인을 입력합니다.

   예를 들어, 하위 도메인의 소유권을 확인하기 위한 Amplify 콘솔의 DNS 레코드가 _c3e2d7eaf1e656b73f46cd6980fdc0e.example.com인 경우, 이름에 **`_c3e2d7eaf1e656b73f46cd6980fdc0e`**만 입력합니다.

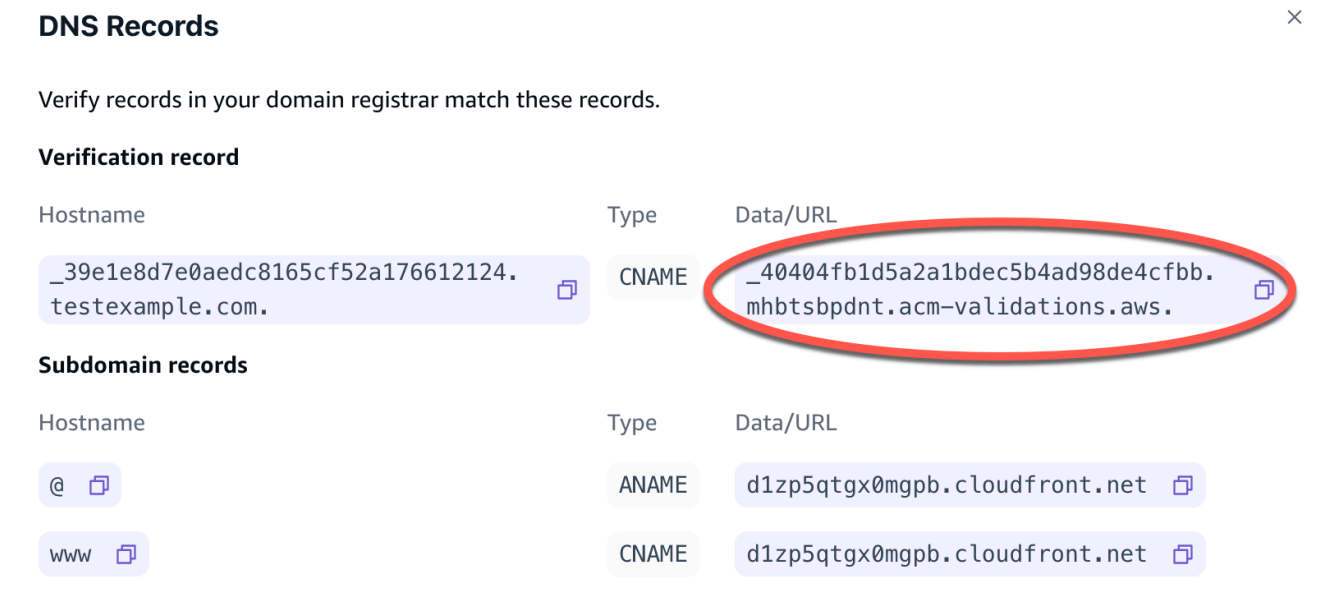
다음 스크린샷은 사용할 확인 레코드의 위치를 보여줍니다.



c. 값에는 ACM 검증 인증서를 입력하십시오.

예를 들어 검증 서버가 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws인 경우, 값에 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws를 입력합니다.

다음 스크린샷은 사용할 ACM 확인 레코드의 위치를 보여줍니다.



d. 저장을 선택합니다.

> **Note**
>
> AWS Certificate Manager (ACM)에서 생성한 기본 Amplify 인증서는 13개월 동안 유효하며 앱이 Amplify로 호스팅되는 한 자동으로 갱신됩니다. CNAME 확인 레코드가 수정 또는 삭제된 경우, Amplify는 인증서를 갱신할 수 없습니다. Amplify 콘솔에서 도메인을 삭제하고 다시 추가해야 합니다.

7. 하위 도메인에 대해 이 단계는 필수가 아닙니다. GoDaddy는 ANAME/ALIAS 레코드를 지원하지 않습니다. ANAME/ALIAS 지원 기능이 없는 DNS 공급자의 경우, DNS를 Amazon Route 53으로 마이그레이션하는 것이 좋습니다. 자세한 내용은 Amazon Route 53을 DNS 서비스로 구성을 참조하십시오.

   현재 공급자를 GoDaddy로 유지하고 루트 도메인을 업데이트하려면 전송을 추가하고 도메인을 전송으로 설정하십시오.

   a. DNS 페이지에서 페이지 상단의 메뉴를 찾아 전송을 선택합니다.
   b. 도메인 섹션에서 전송 추가를 선택합니다.
   c. http://를 선택한 다음 대상 URL에 전송 대상인 하위 도메인의 이름을 입력합니다(예: www.example.com).
   d. 전송 유형에서는 임시(302)를 선택합니다.
   e. 저장을 선택합니다.

도메인의 SSL/TLS 인증서 업데이트

도메인에 사용 중인 SSL/TLS 인증서를 언제든지 변경할 수 있습니다. 예를 들어 관리형 인증서 사용에서 사용자 지정 인증서 사용으로 변경할 수 있습니다. 이는 인증서와 만료 알림을 관리하려는 경우에 유용합니다. 도메인에 사용 중인 사용자 지정 인증서를 변경할 수도 있습니다. SSL 인증서를 변경해도 활성 도메인에 가동 중지가 발생하지 않습니다. 인증서에 대한 자세한 내용은 SSL/TLS 인증서 사용을 참조하세요.

도메인에 사용 중인 인증서 유형 또는 사용자 지정 인증서를 업데이트하려면 다음 절차를 사용합니다.

도메인의 인증서를 업데이트하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 업데이트할 앱을 선택합니다.

3. 탐색 창에서 호스팅, 사용자 도메인을 선택합니다.
4. 사용자 지정 도메인 페이지에서 도메인 구성을 선택합니다.
5. 도메인의 세부 정보 페이지에서 사용자 지정 SSL 인증서 섹션을 찾습니다. 인증서를 업데이트하는 절차는 변경 유형에 따라 달라집니다.
   - 사용자 지정 인증서에서 기본 Amplify 관리형 인증서로 변경하려면
     - Amplify 관리형 인증서를 선택합니다.
   - 관리형 인증서에서 사용자 지정 인증서로 변경하려면
     a. 사용자 지정 SSL 인증서를 선택합니다.
     b. 목록에서 사용할 인증서를 선택합니다.
   - 한 사용자 지정 인증서를 다른 사용자 지정 인증서로 변경하려면
     - 사용자 지정 SSL 인증서의 목록에서 사용할 새 인증서를 선택합니다.
6. 저장을 선택합니다. 도메인의 상태 세부 정보는 Amplify가 관리형 인증서의 SSL 생성 프로세스 또는 사용자 지정 인증서의 구성 프로세스를 시작했음을 나타냅니다.

하위 도메인 관리

하위 도메인은 도메인 이름 앞에 나타나는 URL의 일부입니다. 예를 들어 www는 www.amazon.com의 하위 도메인이고 aws는 aws.amazon.com의 하위 도메인입니다. 이미 프로덕션 웹사이트가 있는 경우, 하위 도메인만 연결하는 것이 좋습니다. 하위 도메인은 다중 수준일 수도 있습니다. 예를 들어 beta.alpha.example.com에는 다중 수준 하위 도메인 beta.alpha가 있습니다.

하위 도메인만 추가하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 하위 도메인을 추가할 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 사용자 지정 도메인을 선택합니다.
4. 사용자 지정 도메인 페이지에서 도메인 추가를 선택합니다.
5. 루트 도메인의 이름을 입력한 다음 도메인 구성을 선택합니다. 예를 들어 도메인의 이름이 https://example.com인 경우 example.com을 입력합니다.
6. 루트 제외를 선택하고 하위 도메인의 이름을 수정합니다. 예를 들어 도메인이 example.com인 경우, 하위 도메인 알파만 추가하도록 수정할 수 있습니다.
7. 도메인 추가를 선택합니다.

다단계 하위 도메인을 추가하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 다단계 하위 도메인을 추가할 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 사용자 지정 도메인을 선택합니다.
4. 사용자 지정 도메인 페이지에서 도메인 추가를 선택합니다.
5. 하위 도메인이 있는 도메인의 이름을 입력하고 루트 제외를 선택한 다음 하위 도메인을 수정하여 새 수준을 추가합니다.

   예를 들어 alpha.example.com이라는 도메인이 있고 다단계 하위 도메인 beta.alpha.example.com을 생성하려면 베타를 하위 도메인 값으로 입력합니다.
6. 도메인 추가를 선택합니다.

하위 도메인을 추가 또는 편집하려면

앱에 사용자 지정 도메인을 추가한 후 기존 하위 도메인을 편집하거나 새 하위 도메인을 추가할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 하위 도메인을 관리할 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 사용자 지정 도메인을 선택합니다.
4. 사용자 지정 도메인 페이지에서 도메인 구성을 선택합니다.
5. 하위 도메인 섹션에서 필요에 따라 기존 하위 도메인을 편집할 수 있습니다.
6. (선택 사항) 새 하위 도메인을 추가하려면 새로 추가를 선택합니다.
7. 저장을 선택합니다.

와일드카드 하위 도메인 설정

Amplify 호스팅은 이제 와일드카드 하위 도메인을 지원합니다. 와일드카드 하위 도메인은 기존 하위 도메인과 존재하지 않는 하위 도메인을 애플리케이션의 특정 브랜치로 가리킬 수 있는 포괄적인 하위 도메인입니다. 와일드카드를 사용하여 앱의 모든 하위 도메인을 특정 브랜치에 연결하면 모든 하위 도메인의 앱 사용자에게 동일한 콘텐츠를 제공할 수 있으므로 각 하위 도메인을 개별적으로 구성하지 않아도 됩니다.

와일드카드 하위 도메인을 만들려면 하위 도메인 이름으로 별표(*) 를 지정하십시오. 예를 들어 앱의 특정 브랜치에 와일드카드 하위 도메인을 `*.example.com` 지정하는 경우, example.com으로 끝나는 모든 URL은 브랜치로 라우팅됩니다. 이 경우, `dev.example.com` 및 에 대한 요청은 하위 도메인으로 `prod.example.com` 라우팅됩니다. `*.example.com`

Amplify는 사용자 지정 도메인에 대해서만 와일드카드 하위 도메인을 지원합니다. 기본 `amplifyapp.com` 도메인에서는 이 기능을 사용할 수 없습니다.

Wildcard 하위 도메인에 적용되는 요구 사항은 다음과 같습니다.

- 하위 도메인 이름은 별표(*) 로만 지정해야 합니다.
- 와일드카드를 사용하여 하위 도메인 이름의 일부를 대체할 수 없습니다(예: *domain.example.com).
- 도메인 이름의 중간에 있는 하위 도메인은 바꿀 수 없습니다(예: subdomain.*.example.com).
- 기본적으로 모든 Amplify 프로비저닝 인증서는 사용자 지정 도메인의 모든 하위 도메인을 포함합니다.

와일드카드 하위 도메인을 추가 또는 삭제하려면

앱에 커스텀 도메인을 추가한 후 앱 브랜치에 와일드카드 하위 도메인을 추가할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 Amplify Hosting 콘솔을 엽니다.
2. 와일드카드 하위 도메인을 관리할 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 사용자 지정 도메인을 선택합니다.
4. 사용자 지정 도메인 페이지에서 도메인 구성을 선택합니다.
5. 하위 도메인 섹션에서 와일드카드 하위 도메인을 추가하거나 삭제할 수 있습니다.
   - 새 Wildcard 하위 도메인을 추가하려면
     a. 새로 추가를 선택합니다.
     b. 하위 도메인에 대해 *****을 입력합니다.
     c. 앱 브랜치의 경우, 목록에서 브랜치 이름을 선택합니다.
     d. 저장을 선택합니다.
   - 와일드카드 하위 도메인을 삭제하려면
     a. 하위 도메인 이름 옆의 제거를 선택합니다. 명시적으로 구성되지 않은 하위 도메인으로의 트래픽이 중지되고 Amplify Hosting은 해당 요청에 404 상태 코드를 반환합니다.
     b. 저장을 선택합니다.

Amazon Route 53 사용자 지정 도메인을 위한 자동 하위 도메인 설정

앱이 Route 53의 사용자 지정 도메인에 연결되면 Amplify를 사용하면 새로 연결된 브랜치의 하위 도메인을 자동으로 생성할 수 있습니다. 예를 들어 dev 브랜치를 연결하면 Amplify는 dev.exampledomain.com을 자동으로 생성할 수 있습니다. 브랜치를 삭제하면 연결된 모든 하위 도메인이 자동으로 삭제됩니다.

새로 연결된 브랜치에 자동 하위 도메인 생성을 설정하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. Route 53에서 관리되는 사용자 지정 도메인에 연결된 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 사용자 지정 도메인을 선택합니다.
4. 사용자 지정 도메인 페이지에서 도메인 구성을 선택합니다.
5. 자동 하위 도메인 생성 섹션에서 기능을 켭니다.

> **Note**
>
> 이 기능은 루트 도메인(예: exampledomain.com)에서만 사용할 수 있습니다. 도메인이 이미 dev.exampledomain.com과 같은 하위 도메인인 경우에는 Amplify 콘솔에 이 확인란이 표시되지 않습니다.

하위 도메인이 포함된 웹 미리 보기

위 지침에 따라 자동 하위 도메인 생성을 활성화하면 자동으로 생성된 하위 도메인을 통해 앱의 풀 요청 웹 미리 보기에도 액세스할 수 있습니다. pull 요청이 종료되면 연결된 브랜치 및 하위 도메인이 자동으로 삭제됩니다. pull 요청의 웹 미리 보기 설정에 대한 자세한 내용은 pull 요청를 위한 웹 미리 보기 단원을 참조하십시오.

사용자 지정 도메인 문제 해결

AWS Amplify 콘솔에서 앱에 사용자 지정 도메인을 추가할 때 문제가 발생하는 경우 Amplify 문제 해결 장의 사용자 지정 도메인 문제 해결 섹션을 참조하세요. 여기에서 문제 해결 방법을 찾을 수 없다면 지원에 문의하세요. 자세한 내용을 알아보려면 AWS Support 사용 설명서의 지원 사례 만들기를 참조하세요.

Amplify 호스팅 사이트에 대한 방화벽 지원

Amplify 호스팅 사이트에 대한 방화벽 지원을 통해와 직접 통합되어 웹 애플리케이션을 보호할 수 있습니다 AWS WAF.를 AWS WAF 사용하면 정의한 사용자 지정 가능한 웹 보안 규칙 및 조건을 기반으로 웹 요청을 허용, 차단 또는 모니터링(계산)하는 웹 액세스 제어 목록(웹 ACL)이라는 규칙 세트를 구성할 수 있습니다. Amplify 앱을와 통합하면 앱에서 허용하는 HTTP 트래픽에 대한 제어 및 가시성을 AWS WAF높일 수 있습니다. 자세한 내용은 AWS WAF 개발자 안내서의 AWS WAF 작동 방식을 AWS WAF참조하세요.

Amplify Hosting이 작동하는 모든 AWS 리전 에서 방화벽 지원을 사용할 수 있습니다. 이 통합은 CloudFront와 유사한 AWS WAF 글로벌 리소스에 속합니다. 웹 ACLs은 여러 Amplify Hosting 앱에 연결할 수 있지만 동일한 리전에 있어야 합니다.

AWS WAF 를 사용하여 SQL 삽입 및 교차 사이트 스크립팅과 같은 일반적인 웹 악용으로부터 Amplify 앱을 보호할 수 있습니다. 이는 앱의 가용성과 성능에 영향을 미치거나, 보안을 손상시키거나, 과도한 리소스를 소비할 수 있습니다. 예를 들어 지정된 IP 주소 범위의 요청, CIDR 블록의 요청, 특정 국가 또는 리전에서 시작된 요청 또는 예상치 못한 SQL 코드 또는 스크립팅이 포함된 요청을 허용하거나 차단하는 규칙을 생성할 수 있습니다.

HTTP 헤더, 메서드, 쿼리 문자열, URI 및 요청 본문(처음 8KB로 제한)에 지정된 문자열이나 정규식 패턴과 일치하는 규칙을 만들 수도 있습니다. 또한 특정 사용자 에이전트, 봇 및 콘텐츠 스크레이퍼의 이벤트를 차단하는 규칙을 생성할 수 있습니다. 예를 들어, 비율 기반 규칙을 사용하여 연속적으로 업데이트되는 이후 5분 동안 각 클라이언트 IP에서 허용하는 웹 요청 수를 지정할 수 있습니다.

지원되는 규칙 유형과 추가 AWS WAF 기능에 대한 자세한 내용은 AWS WAF 개발자 안내서 및 AWS WAF API 참조를 참조하세요.

> **Important**
>
> 보안은 AWS 와 사용자 간의 공동 책임입니다. AWS WAF 는 모든 인터넷 보안 문제에 대한 솔루션이 아니므로 보안 및 규정 준수 목표를 충족하도록 구성해야 합니다. 사용 시 공동 책임 모델을 적용하는 방법을 이해하려면 서비스 사용 시 보안을 AWS WAF참조하세요. AWS WAF

주제

- 에서 Amplify 애플리케이션 AWS WAF 활성화 AWS Management Console
- Amplify 애플리케이션에서 웹 ACL 연결 해제

- 를 사용하여 Amplify 애플리케이션 AWS WAF 활성화 AWS CDK
- Amplify가와 통합되는 방법 AWS WAF
- Amplify 애플리케이션의 방화벽 요금

에서 Amplify 애플리케이션 AWS WAF 활성화 AWS Management Console

Amplify 콘솔 또는 콘솔에서 Amplify 앱에 대한 AWS WAF 보호를 활성화할 수 있습니다 AWS WAF .

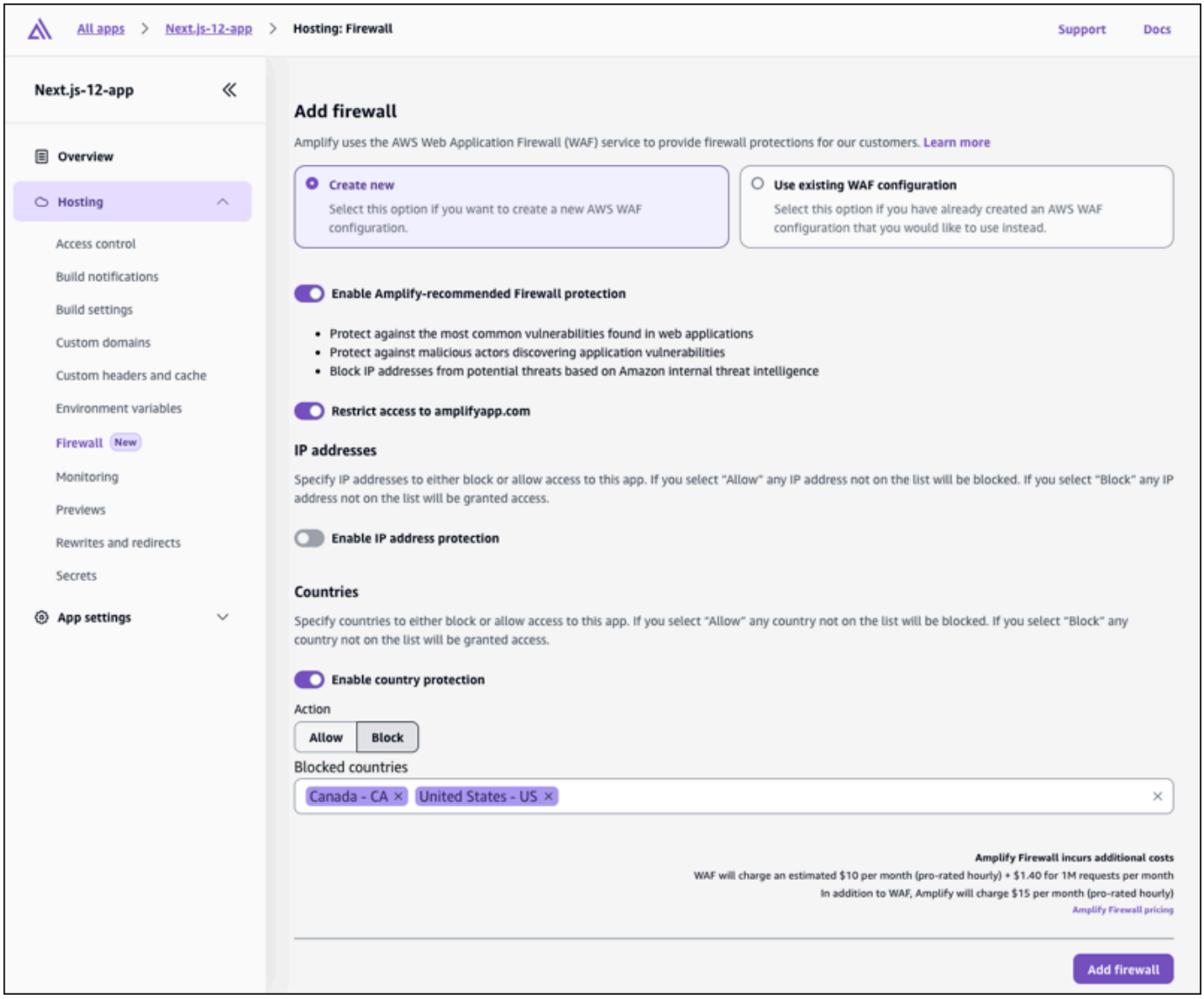
- Amplify 콘솔 - Amplify 콘솔에서 AWS WAF 웹 ACL을 앱에 연결하여 기존 Amplify 앱에 대한 방화벽 기능을 활성화할 수 있습니다. 원클릭 보호를 사용하여 대부분의 앱에서 모범 사례로 간주되는 사전 구성된 규칙을 사용하여 웹 ACL을 생성합니다. IP 주소 및 국가별로 액세스를 사용자 지정할 수 있습니다. 이 섹션의 지침에서는 원클릭 보호 설정에 대해 설명합니다.
- AWS WAF 콘솔 - AWS WAF 콘솔에서 생성하거나 AWS WAF APIs를 사용하여 미리 구성된 웹 ACL을 사용합니다. 글로벌(CloudFront) 리전에서 Amplify 앱과 연결할 웹 ACLs을 생성해야 합니다. 리전 웹 ACLs은에 이미 있을 수 AWS 계정있지만 Amplify와 호환되지 않습니다. 시작하기 지침은 AWS WAF 개발자 안내서의 설정 AWS WAF 및 해당 구성 요소를 참조하세요.

Amplify 콘솔에서 기존 앱에 AWS WAF 대해를 활성화하려면 다음 절차를 사용합니다.

기존 Amplify 앱 AWS WAF 에 대해 활성화

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/amplify/ Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 방화벽 기능을 활성화할 배포된 앱의 이름을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 방화벽을 선택합니다.

   다음 스크린샷은 Amplify 콘솔에서 방화벽 추가 페이지로 이동하는 방법을 보여줍니다.



4. 방화벽 추가 페이지에서 작업은 새 구성을 생성할지 아니면 기존 AWS WAF 구성을 사용할지에 따라 달라집니다.

   - 새 AWS WAF 구성을 생성합니다.

     a. 새로 생성을 선택합니다.

     b. 선택적으로 다음 구성 중 하나를 활성화합니다.

        i. Amplify 권장 방화벽 보호 활성화를 켭니다.

        ii. 기본 Amplify 도메인에서 앱에 대한 액세스를 방지하려면 amplifyapp.com 액세스 제한을 켭니다.

        iii. IP 주소의 경우 IP 주소 보호 활성화를 켭니다.

A. 작업에서 액세스 권한이 있고 다른 모든 IP 주소가 차단되는 IP 주소를 지정하려면 허용을 선택합니다. 차단될 IP 주소와 다른 모든 IP 주소에 대한 액세스 권한을 지정하려면 차단을 선택합니다.

B. IP 버전에서 IPV4 또는 IPV6을 선택합니다.

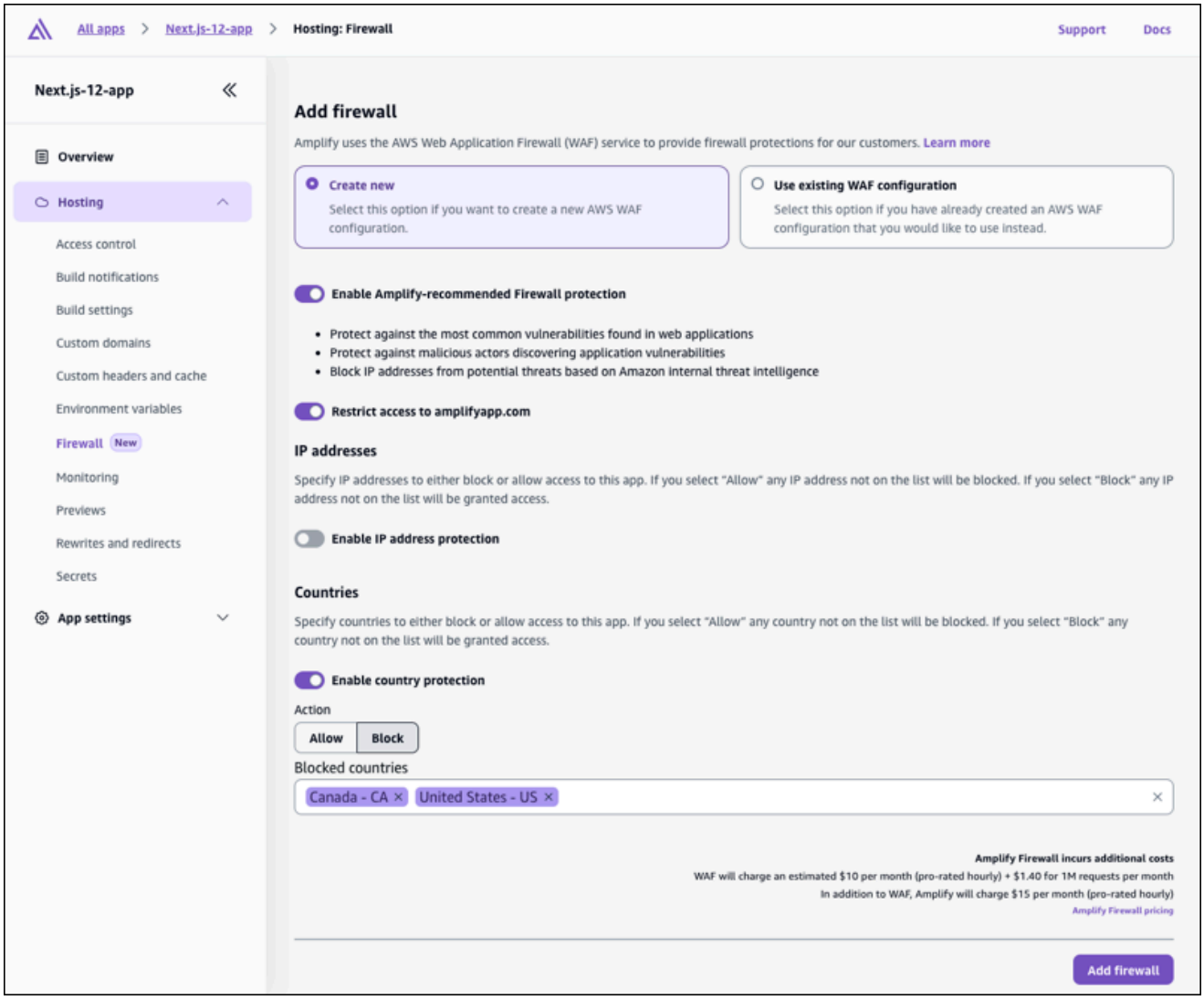
C. IP 주소 텍스트 상자에 허용된 IP 주소 또는 차단된 IP 주소를 CIDR 형식으로 한 줄에 하나씩 입력합니다.

iv. 국가의 경우 국가 보호 활성화를 켭니다.

A. 작업에서 액세스 권한이 있는 국가를 지정하고 다른 모든 국가는 차단되도록 하려면 허용을 선택합니다. 차단될 국가와 다른 모든 국가에 액세스할 수 있는 국가를 지정하려면 차단을 선택합니다.

B. 국가의 경우 목록에서 허용된 국가 또는 차단된 국가를 선택합니다.

다음 스크린샷은 앱에 대해 새 AWS WAF 구성을 활성화하는 방법을 보여줍니다.



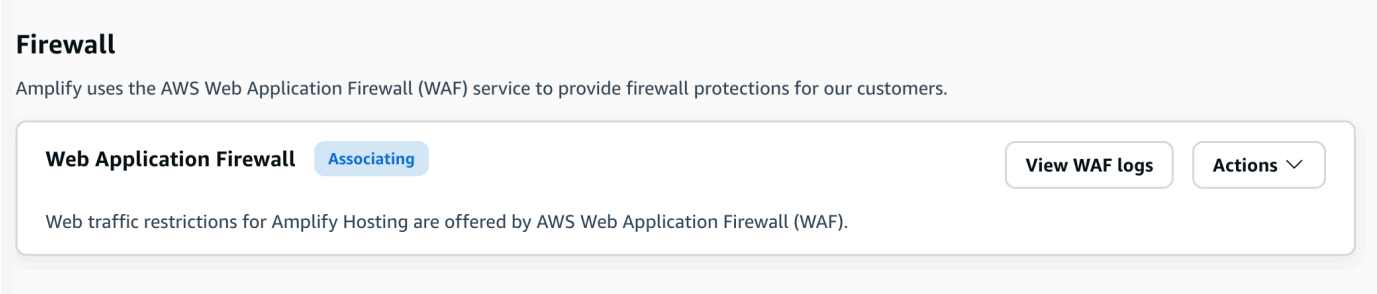
- 기존 AWS WAF 구성을 사용합니다.

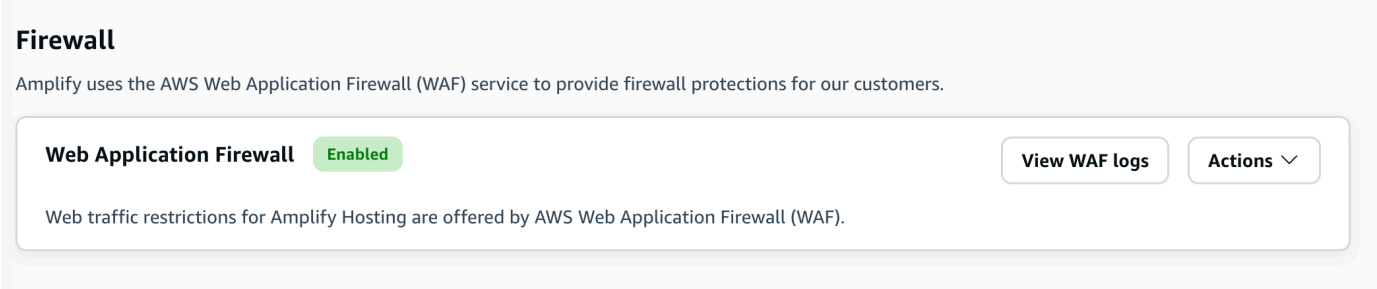
  a. 기존 AWS WAF 구성 사용을 선택합니다.

  b. 의 AWS WAF 에 있는 웹 ACLs 목록에서 저장된 구성을 선택합니다 AWS 계정. Amplify 앱과 연결하는 웹 ACL은 글로벌(CloudFront) 리전에서 생성해야 합니다. 리전 웹 ACLs 은에 이미 있을 수 AWS 계정있지만 Amplify와 호환되지 않습니다.

5. 방화벽 추가를 선택합니다.
6. 방화벽 페이지에 연결 중 상태가 표시되어 AWS WAF 설정이 전파되고 있음을 나타냅니다. 프로세스가 완료되면 상태가 활성화됨으로 변경됩니다.

   다음 스크린샷은 Amplify 콘솔의 방화벽 진행 상태를 보여 주며, AWS WAF 구성이 연결 중 및 활성화됨임을 나타냅니다.





Amplify 애플리케이션에서 웹 ACL 연결 해제

Amplify 앱과 연결된 웹 ACL은 삭제할 수 없습니다. 먼저 Amplify 콘솔의 앱에서 웹 ACL의 연결을 해제해야 합니다. 그런 다음 AWS WAF 콘솔에서 삭제할 수 있습니다.

Amplify 앱에서 웹 ACL을 연결 해제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/amplify/ Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 웹 ACL의 연결을 해제할 앱의 이름을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 방화벽을 선택합니다.
4. 방화벽 페이지에서 작업을 선택한 다음 방화벽 연결 해제를 선택합니다.
5. 확인 모달에서를 입력한 **`disassociate`**다음 방화벽 연결 해제를 선택합니다.
6. 방화벽 페이지에 연결 해제 상태가 표시되어 AWS WAF 설정이 전파되고 있음을 나타냅니다.

   프로세스가 완료되면 AWS WAF 콘솔에서 웹 ACL을 삭제할 수 있습니다.

를 사용하여 Amplify 애플리케이션 AWS WAF 활성화 AWS CDK

를 사용하여 Amplify 애플리케이션에 AWS WAF 대해를 활성화 AWS 클라우드 개발 키트 (AWS CDK) 할 수 있습니다. CDK 사용에 대한 자세한 내용은 AWS 클라우드 개발 키트 (AWS CDK) 개발자 안내서의 CDK란 무엇입니까?를 참조하세요.

다음 TypeScript 코드 예제에서는 두 개의 CDK 스택으로 AWS CDK 앱을 생성하는 방법을 보여줍니다. 하나는 Amplify용이고 다른 하나는 용입니다 AWS WAF. AWS WAF 스택은 미국 동부(버지니아 북부)(us-east-1) 리전에 배포해야 합니다. Amplify 애플리케이션 스택은 다른 리전에 배포할 수 있습니다. 글로벌(CloudFront) 리전에서 Amplify 앱과 연결할 웹 ACL을 생성해야 합니다. 리전 웹 ACLs은에 이미 있을 수 AWS 계정있지만 Amplify와 호환되지 않습니다.

```
import * as cdk from "aws-cdk-lib";
import { Construct } from "constructs";
import * as wafv2 from "aws-cdk-lib/aws-wafv2";
import * as amplify from "aws-cdk-lib/aws-amplify";

interface WafStackProps extends cdk.StackProps {
  appArn: string;
}

export class AmplifyStack extends cdk.Stack {
  public readonly appArn: string;
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);
    const amplifyApp = new amplify.CfnApp(this, "AmplifyApp", {
      name: "MyApp",
    });
    this.appArn = amplifyApp.attrArn;
  }
}

export class WAFStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props: WafStackProps) {
    super(scope, id, props);
    const webAcl = new wafv2.CfnWebACL(this, "WebACL", {
      defaultAction: { allow: {} },
      scope: "CLOUDFRONT",
      rules: [
        // Add your own rules here.
      ],
      visibilityConfig: {
        cloudWatchMetricsEnabled: true,
        metricName: "my-metric-name",
        sampledRequestsEnabled: true,
      },
    });

    new wafv2.CfnWebACLAssociation(this, "WebACLAssociation", {
```

```
      resourceArn: props.appArn,
      webAclArn: webAcl.attrArn,
    });
  }
}

const app = new cdk.App();

// Create AmplifyStack in your desired Region.
const amplifyStack = new AmplifyStack(app, 'AmplifyStack', {
  env: { region: 'us-west-2' },
});

// Create WAFStack in IAD region, passing appArn from AmplifyStack.
new WAFStack(app, 'WAFStack', {
  env: { region: 'us-east-1' },
  crossRegionReferences: true,

  appArn: amplifyStack.appArn,  // Pass appArn from AmplifyStack.
});
```

Amplify가와 통합되는 방법 AWS WAF

다음 목록은 방화벽 지원이 AWS WAF 와 통합되는 방법과 웹 ACLs을 생성하고 이를 Amplify 앱과 연결할 때 고려해야 할 제약 조건에 대한 구체적인 세부 정보를 제공합니다.

- 모든 유형의 Amplify 앱 AWS WAF 에 대해를 활성화할 수 있습니다. 여기에는 지원되는 프레임워크, 서버 측 렌더링(SSR) 앱 및 완전 정적 사이트가 포함됩니다. AWS WAF 는 Amplify Gen 1 및 Gen 2 앱에서 지원됩니다.
- 글로벌(CloudFront) 리전에서 Amplify 앱과 연결할 웹 ACLs을 생성해야 합니다. 리전 웹 ACLs은에 이미 있을 수 AWS 계정있지만 Amplify와 호환되지 않습니다.
- 웹 ACL과 Amplify 앱은 동일한에서 생성해야 합니다 AWS 계정. AWS Firewall Manager 를 사용하여 규칙을 복제 AWS 계정하고 조직 AWS WAF 규칙을 중앙 집중화하고 여러에 분산하는 작업을 간소화할 수 있습니다 AWS 계정. 자세한 내용은 AWS WAF 개발자 안내서의 AWS Firewall Manager 섹션을 참조하세요.
- 동일한의 여러 Amplify 앱에서 동일한 웹 ACL을 공유할 수 있습니다 AWS 계정. 모든 앱은 동일한 리전에 있어야 합니다.
- 웹 ACL을 Amplify 앱과 연결하면 기본적으로 웹 ACL이 앱의 모든 브랜치에 연결됩니다. 새 브랜치를 생성하면 웹 ACL이 생성됩니다.

- 웹 ACL을 Amplify 앱에 연결하면 모든 앱의 도메인과 자동으로 연결됩니다. 그러나 호스트 헤더 일치 규칙을 사용하여 단일 도메인 이름에 적용되는 규칙을 구성할 수 있습니다.
- Amplify 앱과 연결된 웹 ACL은 삭제할 수 없습니다. AWS WAF 콘솔에서 웹 ACL을 삭제하기 전에 앱에서 연결을 해제해야 합니다.

Amplify 웹 ACL 리소스 정책

Amplify가 웹 ACL에 액세스할 수 있도록 연결 중에 리소스 정책이 웹 ACL에 연결됩니다. Amplify는이 리소스 정책을 자동으로 구성하지만 AWS WAFV2 GetPermissionPolicy API를 사용하여 볼 수 있습니다. 웹 ACL을 Amplify 앱에 연결하려면 다음 IAM 권한이 필요합니다.

- amplify:AssociateWebACL
- wafv2:AssociateWebACL
- wafv2:PutPermissionPolicy
- wafv2:GetPermissionPolicy

Amplify 애플리케이션의 방화벽 요금

Amplify 애플리케이션에서 구현 AWS WAF 하는 비용은 다음 두 가지 구성 요소를 기반으로 계산됩니다.

- AWS WAF 사용량 - AWS WAF 요금 모델에 따른 사용량 조정에 대한 AWS WAF 요금이 부과됩니다. AWS WAF 요금은 생성한 웹 액세스 제어 목록(웹 ACLs), 웹 ACL당 추가하는 규칙 수, 수신하는 웹 요청 수를 기준으로 합니다. 요금에 대한 자세한 내용은 AWS WAF 요금을 참조하세요.
- Amplify Hosting 통합 비용 - Amplify 애플리케이션에 웹 ACL을 연결할 때 앱당 월 15.00 USD의 요금이 부과됩니다. 이는 시간당 비례 할당으로 계산됩니다.

기능 브랜치 배포 및 팀 워크플로우

Amplify Hosting은 기능별 브랜치 및 GitFlow 워크플로에서 작동하도록 설계되었습니다. Amplify는 리포지토리에서 새 브랜치를 연결할 때마다 Git 브랜치를 사용하여 새 배포를 생성합니다. 첫 번째 브랜치를 연결한 후 추가 기능 브랜치를 생성합니다.

앱에 브랜치를 추가하려면

1. 브랜치를 추가할 앱을 선택합니다.
2. 앱 설정을 선택한 다음 브랜치 설정을 선택합니다.
3. 브랜치 설정 페이지에서 브랜치 추가를 선택합니다.
4. 리포지토리에서 브랜치를 선택합니다.
5. 브랜치 추가를 선택합니다.
6. 앱을 재배포합니다.

브랜치를 추가하면 앱에 https://main.appid.amplifyapp.com 및 https://dev.appid.amplifyapp.com과 같은 Amplify 기본 도메인에서 두 가지 배포를 사용할 수 있습니다. 이는 팀에 따라 다를 수 있지만 일반적으로 기본 브랜치가 릴리스 코드를 추적하며 프로덕션 브랜치입니다. 개발 브랜치는 새 기능을 테스트하는 데 통합 브랜치로 사용됩니다. 이러한 방식으로 베타 테스터는 기본 브랜치 배포와 관련하여 어떠한 프로덕션 최종 사용자에게도 영향을 주지 않고, 개발 브랜치 배포와 관련하여 릴리스되지 않은 기능을 테스트할 수 있습니다.

주제

- 풀 스택 Amplify Gen 2 앱을 사용한 팀 워크플로
- 풀 스택 Amplify Gen 1 앱을 사용한 팀 워크플로
- 패턴 기반 기능 브랜치 배포
- Amplify 구성의 자동 빌드 타임 생성(Gen 1 앱만 해당)
- 조건부 백엔드 빌드(Gen 1 앱만 해당)
- 앱 전체에서 Amplify 백엔드 사용(Gen 1 앱만 해당)

풀 스택 Amplify Gen 2 앱을 사용한 팀 워크플로

AWS Amplify Gen 2는 백엔드를 정의하기 위한 TypeScript 기반 코드 우선 개발자 환경을 도입합니다. Amplify Gen 2 애플리케이션을 사용한 풀 스택 워크플로에 대한 자세한 내용은 Amplify 설명서의 풀스택 워크플로를 참조하세요.

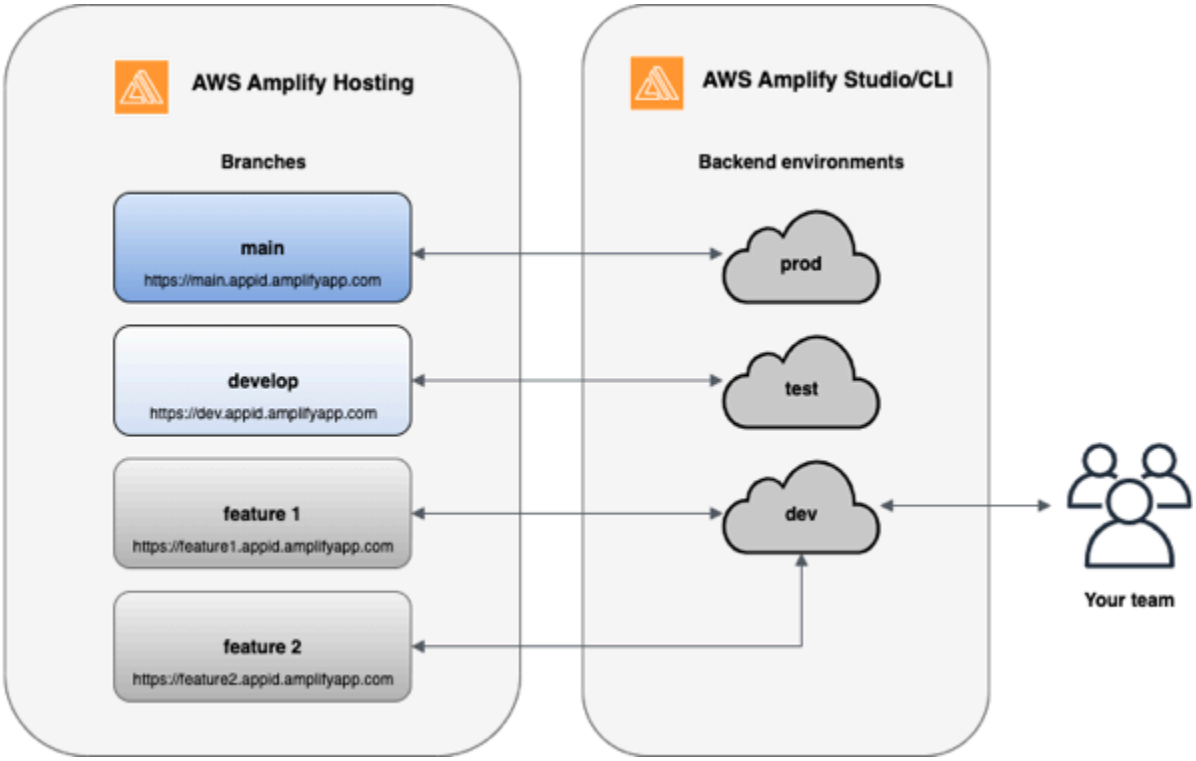
풀 스택 Amplify Gen 1 앱을 사용한 팀 워크플로

기능 브랜치 배포는 프런트엔드와 선택적 백엔드 환경으로 구성됩니다. 프런트엔드는 빌드된 후 글로벌 콘텐츠 배포 네트워크(CDN)에 배포되지만, 백엔드는 Amplify Studio 또는 Amplify CLI에 의해 AWS에 배포됩니다. 이 배포 시나리오를 설정하는 방법을 알아보려면 애플리케이션의 백엔드 빌드 섹션을 참조하세요.

Amplify Hosting은 기능 브랜치 배포와 함께 GraphQL API 및 Lambda 함수와 같은 백엔드 리소스를 지속적으로 배포할 수 있습니다. 다음 브랜칭 모델을 사용하여 Amplify Hosting으로 백엔드 및 프런트엔드를 배포할 수 있습니다.

기능 브랜치 워크플로

- Amplify Studio 또는 Amplify CLI로 prod, 테스트 및 dev 백엔드 환경을 생성합니다.
- prod 백엔드를 기본 브랜치에 매핑하십시오.
- 테스트 백엔드를 개발 브랜치에 매핑합니다.
- 팀 구성원은 dev 백엔드 환경을 사용하여 개별 기능 브랜치를 테스트할 수 있습니다.



1. Amplify CLI를 설치하여 새 Amplify 프로젝트를 초기화합니다.

```
npm install -g @aws-amplify/cli
```

2. 프로젝트에 대해 prod 백엔드 환경을 초기화합니다. 프로젝트가 없는 경우, create-react-app 또는 Gatsby와 같은 부트스트랩 도구를 사용하여 프로젝트를 생성합니다.

```
create-react-app next-unicorn
cd next-unicorn
amplify init
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: prod
...
amplify push
```

3. 테스트 및 dev 백엔드 환경을 추가합니다.

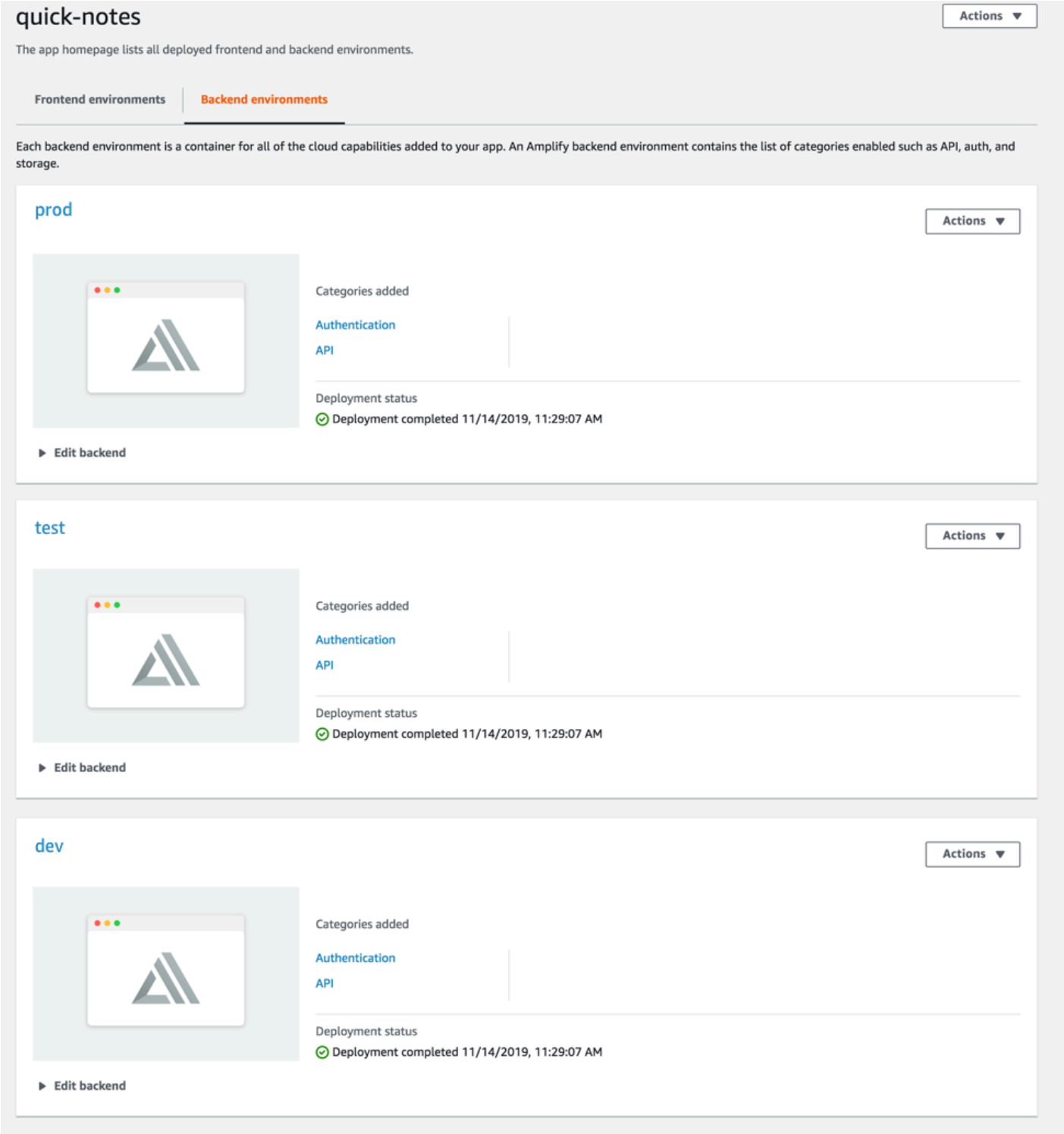
```
amplify env add
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: test
...
amplify push
```

```
amplify env add
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: dev
...
amplify push
```

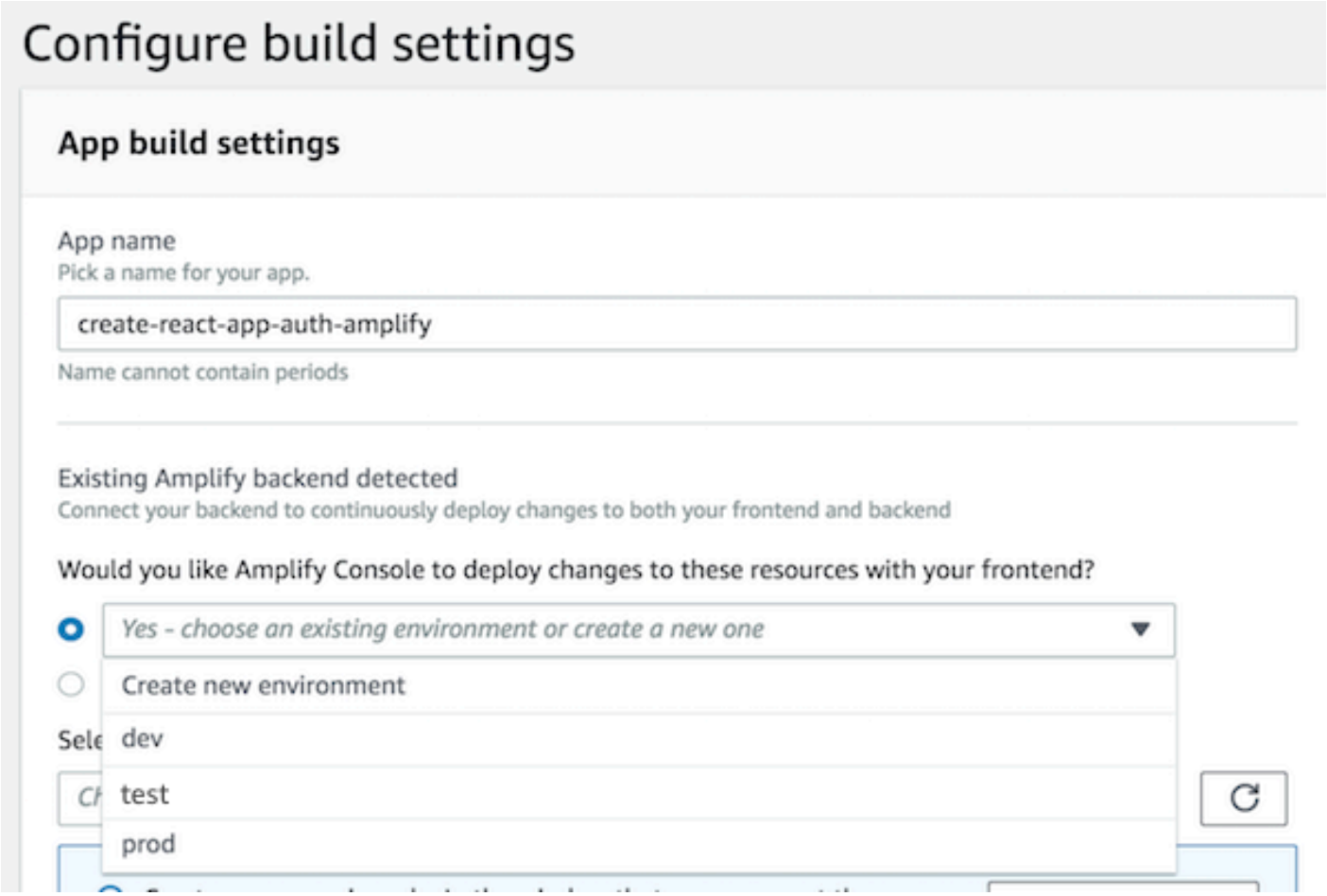
4. 선택한 Git 리포지토리로 코드를 푸시합니다(이 예에서는 사용자가 기본에 푸시한 것으로 가정함).

```
git commit -am 'Added dev, test, and prod environments'
git push origin main
```

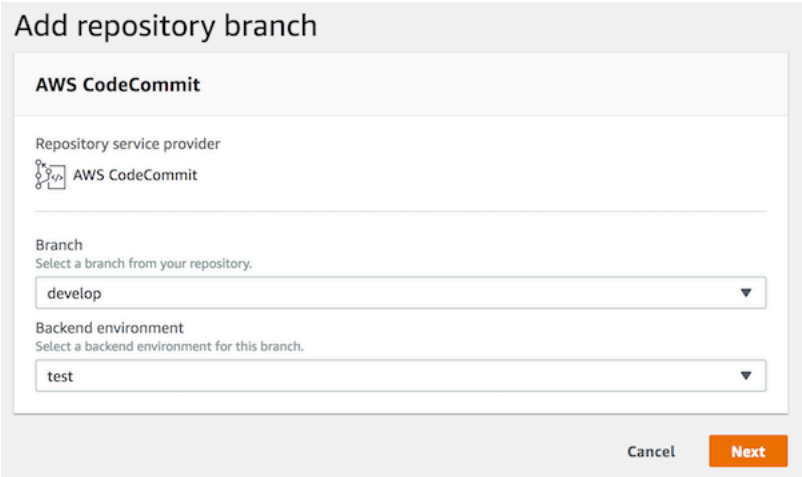
5. 의 Amplify AWS Management Console 를 방문하여 현재 백엔드 환경을 확인합니다. 탐색 경로에서 한 단계 위로 이동하면 백엔드 환경 탭에서 생성된 모든 백엔드 환경 목록을 볼 수 있습니다.



6. 프런트엔드 환경 탭으로 전환하여 리포지토리 제공자와 기본 브랜치를 연결하십시오.
7. 빌드 설정 페이지에서 기존 백엔드 환경을 선택하여 메인 브랜치를 통한 지속적 배포를 설정합니다. 목록에서 prod를 선택하고 Amplify에 서비스 역할을 부여합니다. 저장 및 배포를 선택합니다. 빌드가 완료된 후 https://main.appid.amplifyapp.com에서 사용 가능한 기본 브랜치 배포를 받게 됩니다.



8. Amplify에서 개발 브랜치를 연결합니다(이 시점에 개발 및 기본 브랜치가 동일한 것으로 가정함). test 백엔드 환경을 선택합니다.



9. 이제 Amplify가 설정되었습니다. 기능 브랜치에서 새 기능에 대한 작업을 시작할 수 있습니다. 로컬 워크스테이션에서 dev 백엔드 환경을 사용하여 백엔드 기능을 추가합니다.

```
git checkout -b newinternet
```

```
amplify env checkout dev
amplify add api
...
amplify push
```

10. 기능에 대한 작업을 종료한 후 코드를 커밋하고 풀 요청을 생성하여 내부적으로 검토합니다.

```
git commit -am 'Decentralized internet v0.1'
git push origin newinternet
```

11. 표시될 변경 사항을 미리 보려면 Amplify 콘솔로 이동하여 기능 브랜치를 연결합니다. 참고: 시스템에가 AWS CLI 설치된 경우(Amplify CLI 아님) 터미널에서 직접 브랜치를 연결할 수 있습니다. 애플리케이션 설정 > 일반 > AppARN: arn:aws:amplify:<region>:<region>:apps/<appid>으로 이동하여 appid를 찾아볼 수 있습니다.

```
aws amplify create-branch --app-id <appid> --branch-name <branchname>
aws amplify start-job --app-id <appid> --branch-name <branchname> --job-type RELEASE
```

12. https://newinternet.appid.amplifyapp.com에 액세스하여 기능을 팀 메이트와 공유할 수 있습니다. 모든 것이 순조로워 보이면 PR을 개발 브랜치에 병합합니다.

```
git checkout develop
git merge newinternet
git push
```

13. 이렇게 하면 https://dev.appid.amplifyapp.com에서 브랜치 배포와 함께 Amplify에서 프런트엔드와 백엔드를 업데이트할 빌드가 시작됩니다. 내부 이해관계자가 새 기능을 검토할 수 있도록 이 링크를 내부 이해관계자와 공유할 수 있습니다.

14. Git, Amplify에서 기능 브랜치를 삭제하고, 클라우드에서 백엔드 환경을 제거합니다(언제든지 'amplify env checkout prod' 및 'amplify env add'를 실행하여 새 백엔드 환경을 생성할 수 있음).

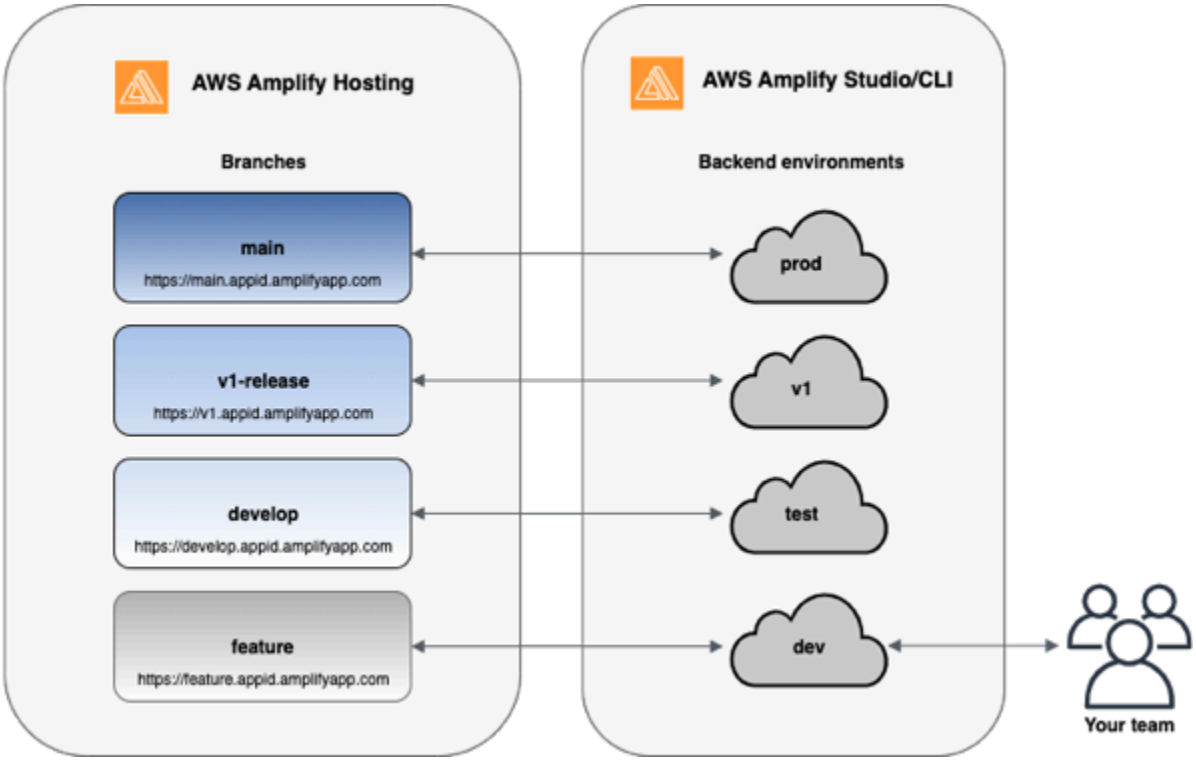
```
git push origin --delete newinternet
aws amplify delete-branch --app-id <appid> --branch-name <branchname>
amplify env remove dev
```

GitFlow 워크플로우

GitFlow는 두 개의 브랜치를 사용하여 프로젝트 이력을 기록합니다. 기본 브랜치는 릴리스 코드만 추적하고 개발 브랜치는 새로운 기능의 통합 브랜치로 사용됩니다. GitFlow는 새로운 개발과 완성된 작업을

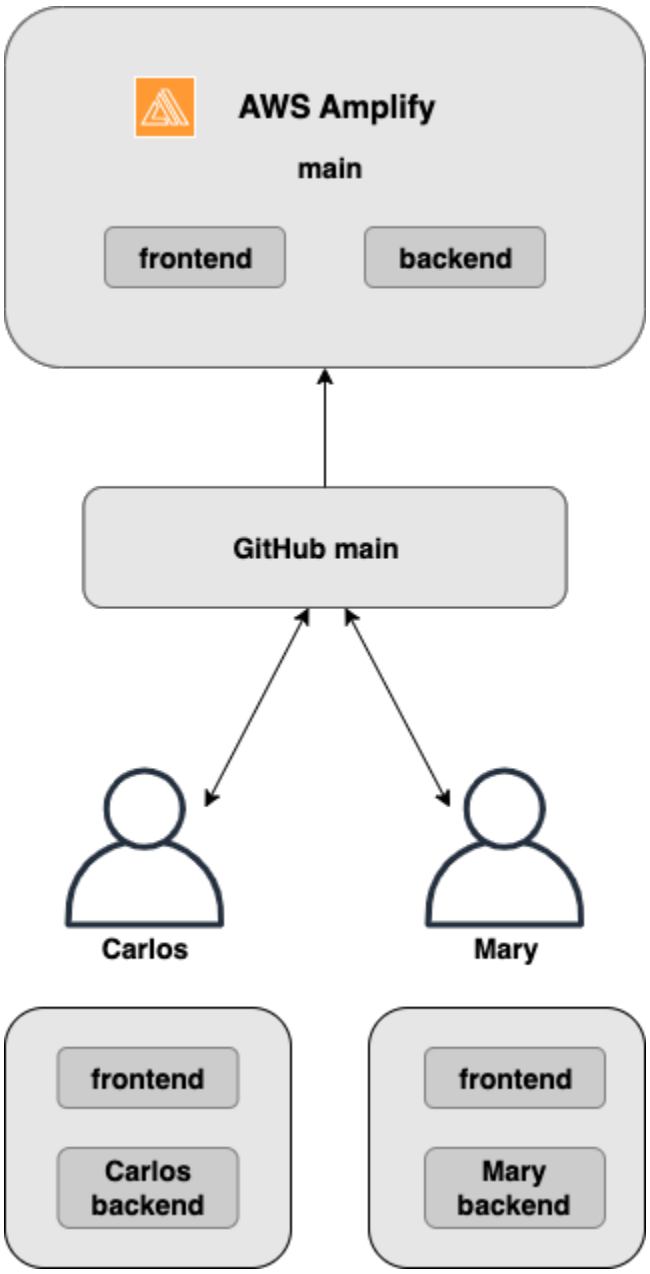
분리하여 병렬 개발을 단순화합니다. 새로운 개발(예: 기능 및 비정기적인 버그 수정)은 기능 브랜치에서 수행됩니다. 개발자가 코드를 릴리스할 준비가 완료되면 기능 브랜치는 통합 개발 브랜치로 병합됩니다. 기본 브랜치에 대한 커밋만 릴리스 브랜치와 핫픽스 브랜치를 병합합니다(긴급 버그 수정).

아래의 다이어그램은 GitFlow를 사용한 권장된 설정을 보여줍니다. 위의 기능 브랜치 워크플로우 섹션에서 설명된 것과 동일한 절차를 따를 수 있습니다.



각 개발자의 샌드박스

- 팀의 각 개발자는 자신의 로컬 컴퓨터와 별도로 클라우드에 샌드박스 환경을 만듭니다 이를 통해 개발자는 다른 팀원의 변경 사항을 덮어쓰지 않고도 서로 격리되어 작업할 수 있습니다.
- Amplify 의 각 브랜치에는 자체 백엔드가 있습니다. 이렇게 하면 팀의 개발자가 직접 로컬 컴퓨터에서 프로덕션 환경으로 백엔드 또는 프런트엔드를 수동으로 푸시하지 않고 Amplify가 Git 리포지토리를 변경 사항 배포를 위한 단일 소스로 사용할 수 있습니다.



1. Amplify CLI를 설치하여 새 Amplify 프로젝트를 초기화합니다.

```
npm install -g @aws-amplify/cli
```

2. 프로젝트에 대해 mary 백엔드 환경을 초기화합니다. 프로젝트가 없는 경우, create-react-app 또는 Gatsby와 같은 부트스트랩 도구를 사용하여 프로젝트를 생성합니다.

```
cd next-unicorn
amplify init
 ? Do you want to use an existing environment? (Y/n): n
```

```
 ? Enter a name for the environment: mary
...
amplify push
```

3. 선택한 Git 리포지토리로 코드를 푸시합니다(이 예에서는 사용자가 기본에 푸시한 것으로 가정함).

```
git commit -am 'Added mary sandbox'
git push origin main
```

4. 리포지토리 > 기본을 Amplify에 연결합니다.
5. Amplify 콘솔은 Amplify CLI에 의해 형성된 백엔드 환경을 감지합니다. 드롭다운에서 새 환경 생성을 선택하고 Amplify에 서비스 역할을 부여합니다. 저장 및 배포를 선택합니다. 빌드가 완료된 후 https://main.appid.amplifyapp.com에서 사용 가능한 마스터 브랜치 배포를 받게 되고 새 백엔드 환경은 브랜치에 연결됩니다.
6. Amplify에서 개발 브랜치를 연결하고(이 시점에 개발 및 기본 브랜치가 동일한 것으로 가정함) 생성을 선택합니다.

패턴 기반 기능 브랜치 배포

배턴 기반 브랜치 배포를 사용하면 특정 패턴을 Amplify에 일치시키는 브랜치를 자동으로 배포할 수 있습니다. 릴리스에 기능 브랜치 또는 GitFlow 워크플로를 사용하는 제품 팀은 이제 '릴리스'로 시작하는 Git 브랜치를 공유 가능한 URL**release****에 자동으로 배포하는 등의 패턴을 정의할 수 있습니다.

1. 앱 설정을 선택한 다음 브랜치 설정을 선택합니다.
2. 브랜치 설정 페이지에서 편집을 선택합니다.
3. 브랜치를 패턴 세트와 일치하는 Amplify에 자동으로 연결하도록 브랜치 자동 감지를 선택합니다.
4. 브랜치 자동 감지 - 패턴 상자에 브랜치를 자동으로 배포하기 위한 패턴을 입력합니다.
   - ***** – 리포지토리의 모든 브랜치를 배포합니다.
   - **release*** – ‘release’라는 단어로 시작하는 모든 브랜치를 배포합니다.
   - **release*/** – ‘release /’ 패턴과 일치하는 모든 브랜치를 배포합니다.
   - 여러 패턴을 쉼표로 구분된 목록으로 지정합니다. 예: **release*, feature***.
5. 브랜치 자동 감지 액세스 제어를 선택하여 자동으로 생성되는 모든 브랜치에 대해 자동 암호 보호를 설정합니다.
6. Amplify 백엔드로 빌드된 Gen 1 애플리케이션의 경우 연결된 모든 브랜치에 대해 새 환경을 생성하거나 기존 백엔드에 대해 모든 브랜치를 가리키도록 선택할 수 있습니다.

7. 저장(Save)을 선택합니다.

사용자 지정 도메인에 연결된 앱을 위한 패턴 기반 기능 분기 배포

Amazon Route 53 사용자 지정 도메인에 연결된 앱에 대해 패턴 기반 기능 분기 배포를 사용할 수 있습니다.

- 패턴 기반 기능 분기 배포 지침은 Amazon Route 53 사용자 지정 도메인을 위한 자동 하위 도메인 설정 단원을 참조하십시오.
- Route 53에서 관리되는 사용자 지정 도메인에 Amplify 앱을 연결하는 방법에 대한 지침은 Amazon Route 53에서 관리하는 사용자 지정 도메인 추가 단원을 참조하십시오.
- Route 53 사용에 대한 자세한 내용은 Amazon Route 53이란을 참조하십시오.

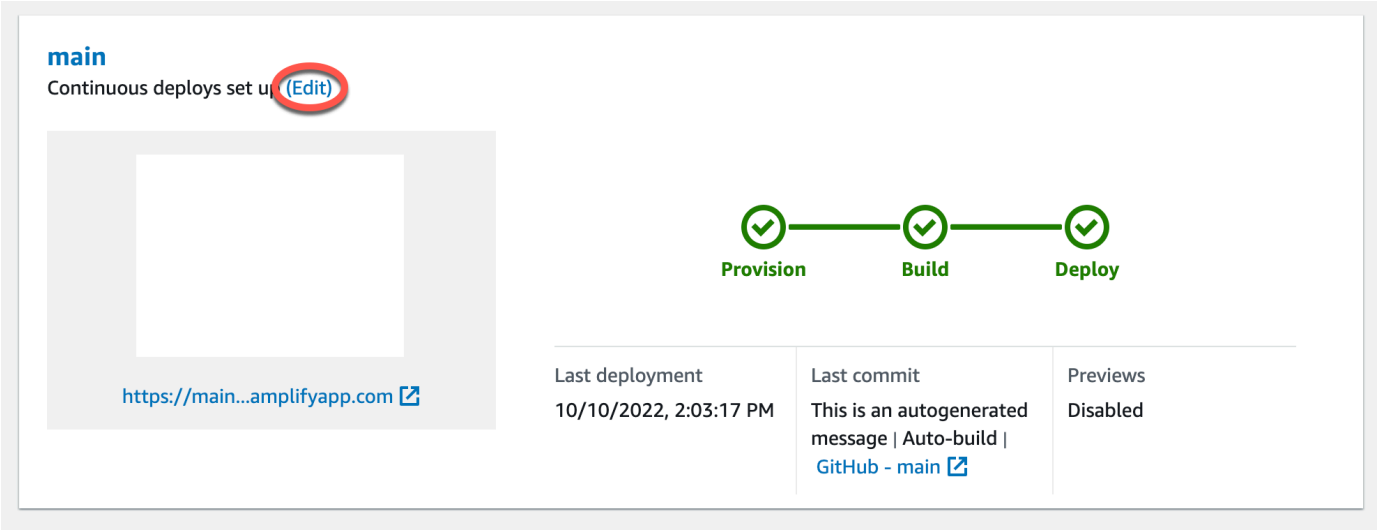
Amplify 구성의 자동 빌드 타임 생성(Gen 1 앱만 해당)

> **Note**
>
> 이 섹션의 정보는 Gen 1 앱에만 적용됩니다. Gen 2 앱의 기능 브랜치에서 인프라 및 애플리케이션 코드 변경 사항을 자동으로 배포하려면 Amplify 설명서의 풀 스택 브랜치 배포를 참조하세요.

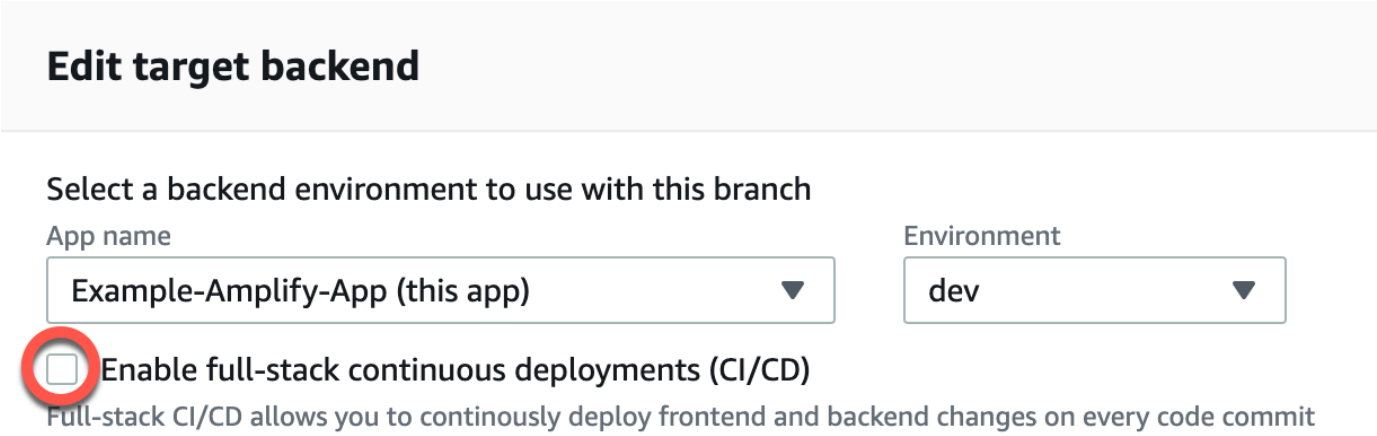
Amplify는 Gen 1 앱에 대해 Amplify 구성 `aws-exports.js` 파일의 자동 빌드 타임 생성을 지원합니다. 풀 스택 CI/CD 배포를 끄면 앱이 `aws-exports.js` 파일을 자동 생성하고 빌드 시 백엔드가 업데이트되지 않도록 할 수 있습니다.

빌드 시 **`aws-exports.js`**를 자동 생성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 편집할 앱을 선택합니다.
3. 호스팅 환경 탭을 선택합니다.
4. 편집할 브랜치를 찾아 편집을 선택합니다.



5. 대상 백엔드 편집 페이지에서 풀 스택 연속 배포(CI/CD)활성화를 선택 취소하여 이 백엔드의 풀 스택 CI/CD를 끕니다.



6. 기존 서비스 역할을 선택하여 Amplify에 앱 백엔드를 변경하는 데 필요한 권한을 부여합니다. 서비스 역할을 생성해야 하는 경우, 새 역할 생성을 선택합니다. 서비스 역할 생성에 대한 자세한 내용은 백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가을 참조하십시오.
7. 저장(Save)을 선택합니다. Amplify는 다음에 앱을 구축할 때 이러한 변경 사항을 적용합니다.

조건부 백엔드 빌드(Gen 1 앱만 해당)

> **Note**
>
> 이 섹션의 정보는 Gen 1 앱에만 적용됩니다. Amplify Gen 2는 TypeScript 기반 코드 우선 개발자 경험을 도입했습니다. 따라서 Gen 2 백엔드에는 이 기능이 필요하지 않습니다.

Amplify는 Gen 1 앱의 모든 브랜치에서 조건부 백엔드 빌드를 지원합니다. 조건부 백엔드 빌드를 구성하려면 `AMPLIFY_DIFF_BACKEND` 환경 변수를 `true`으로 설정합니다. 조건부 백엔드 빌드를 활성화하면 프런트엔드만 변경하는 빌드의 속도를 높이는 데 도움이 됩니다.

diff 기반 백엔드 빌드를 활성화하면 Amplify는 각 빌드를 시작할 때 리포지토리의 `amplify` 폴더에서 diff 실행을 시도합니다. Amplify에서 차이점을 발견하지 못하면 백엔드 빌드 단계를 건너뛰고 백엔드 리소스를 업데이트하지 않습니다. 프로젝트의 리포지토리에 `amplify` 폴더가 없는 경우, Amplify는 `AMPLIFY_DIFF_BACKEND` 환경 변수 값을 무시합니다. `AMPLIFY_DIFF_BACKEND`환경 변수 설정에 대한 지침은 을 참조하십시오. Gen 1 앱을 위한 diff 기반 백엔드 빌드 구성

현재 백엔드 단계의 빌드 설정에 사용자 지정 명령이 지정되어 있는 경우, 조건부 백엔드 빌드는 작동하지 않습니다. 이러한 사용자 지정 명령을 실행하려면 앱의 `amplify.yml` 파일에 있는 빌드 설정의 프런트엔드 단계로 해당 명령을 이동해야 합니다. `amplify.yml` 파일을 업데이트하는 방법에 대한 자세한 내용은 빌드 사양 참조을 참조하십시오.

앱 전체에서 Amplify 백엔드 사용(Gen 1 앱만 해당)

> **Note**
>
> 이 섹션의 정보는 Gen 1 앱에만 적용됩니다. Gen 2 앱에 대한 백엔드 리소스를 공유하려면 Amplify 설명서의 브랜치 간에 리소스 공유를 참조하세요.

Amplify를 사용하면 특정 리전의 모든 Gen 1 앱에서 기존 백엔드 환경을 재사용할 수 있습니다. 새 앱을 만들거나, 새 브랜치를 기존 앱에 연결하거나, 다른 백엔드 환경을 가리키도록 기존 프런트엔드를 업데이트할 때 이 작업을 수행할 수 있습니다.

새 앱 생성 시 백엔드 재사용

새 Amplify 앱을 만들 때 백엔드를 재사용하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 이 예를 활용하려면 다음과 같이 하십시오.
   a. 서비스 탐색 창에서 모든 앱을 선택합니다.
   b. 새 앱, 앱 빌드를 선택합니다.
   c. 앱의 이름(예: **Example-Amplify-App**)을 입력합니다.
   d. 배포 확인을 선택합니다.
3. 프런트엔드를 새 백엔드에 연결하려면 호스팅 환경 탭을 선택합니다.
4. Git 공급자를 선택하고 브랜치 연결을 선택합니다.
5. 리포지토리 브랜치 추가 페이지에서 최근 업데이트된 리포지토리의 경우, 리포지토리 이름을 선택합니다. 브랜치의 경우, 리포지토리에서 연결할 브랜치를 선택합니다.
6. 빌드 설정 페이지에서 다음 작업을 수행하십시오.
   a. 앱 이름에서 백엔드 환경을 추가하는 데 사용할 앱을 선택합니다. 현재 앱 또는 현재 지역의 다른 앱을 선택할 수 있습니다.
   b. 환경에서 추가할 백엔드 환경의 이름을 선택합니다. 기존의 환경을 사용하거나 새 환경을 생성할 수 있습니다.
   c. 기본적으로 풀 스택 CI/CD는 꺼져 있습니다. 풀 스택 CI/CD를 끄면 앱이 풀 전용 모드로 실행됩니다. 빌드 시 Amplify는 백엔드 환경을 수정하지 않고 `aws-exports.js` 파일만 자동으로 생성합니다.
   d. 기존 서비스 역할을 선택하여 Amplify에 앱 백엔드를 변경하는 데 필요한 권한을 부여합니다. 서비스 역할을 생성해야 하는 경우, 새 역할 생성을 선택합니다. 서비스 역할 생성에 대한 자세한 내용은 백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가을 참조하십시오.
   e. Next(다음)를 선택합니다.
7. 저장 및 배포를 선택합니다.

브랜치를 기존 앱에 연결할 때 백엔드를 재사용하십시오.

브랜치를 기존 Amplify 앱에 연결할 때 백엔드를 재사용하려면

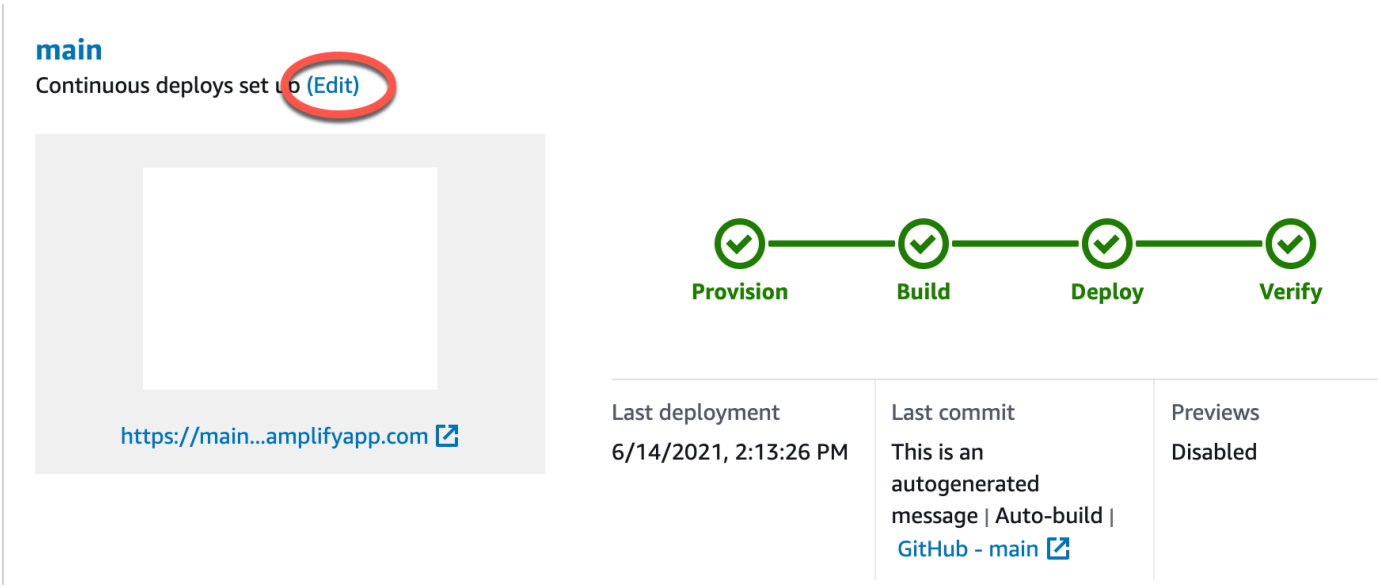
1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.

2. 새 브랜치를 연결할 앱을 선택합니다.
3. 탐색 창에서 앱 설정, 일반을 선택합니다.
4. 브랜치 섹션에서 브랜치 연결을 선택합니다.
5. 리포지토리 브랜치 추가 페이지에서 브랜치의 경우, 리포지토리에서 연결할 브랜치를 선택합니다.
6. 앱 이름에서 백엔드 환경을 추가하는 데 사용할 앱을 선택합니다. 현재 앱 또는 현재 지역의 다른 앱을 선택할 수 있습니다.
7. 환경에서 추가할 백엔드 환경의 이름을 선택합니다. 기존의 환경을 사용하거나 새 환경을 생성할 수 있습니다.
8. Amplify에 앱 백엔드를 변경하는 데 필요한 권한을 부여하기 위해 서비스 역할을 설정해야 하는 경우, 콘솔에 이 작업을 수행하라는 메시지가 표시됩니다. 서비스 역할 생성에 대한 자세한 내용은 백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가을 참조하십시오.
9. 기본적으로 풀스택 CI/CD는 꺼져 있습니다. 풀 스택 CI/CD를 끄면 앱이 풀 전용 모드로 실행됩니다. 빌드 시 Amplify는 백엔드 환경을 수정하지 않고 `aws-exports.js` 파일만 자동으로 생성합니다.
10. Next(다음)를 선택합니다.
11. 저장 및 배포를 선택합니다.

다른 백엔드를 가리키도록 기존 프론트엔드를 편집합니다.

다른 백엔드를 가리키도록 프런트엔드 Amplify 앱을 편집하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 백엔드를 편집할 앱을 선택합니다.
3. 호스팅 환경 탭을 선택합니다.
4. 편집할 브랜치를 찾아 편집을 선택합니다.



5. 이 브랜치와 함께 사용할 백엔드 환경 선택 페이지에서 앱 이름으로 백엔드 환경을 편집하려는 프런트엔드 앱을 선택합니다. 현재 앱 또는 현재 지역의 다른 앱을 선택할 수 있습니다.
6. 백엔드 환경의 경우, 추가할 백엔드 환경의 이름을 선택합니다.
7. 기본적으로 풀스택 CI/CD가 활성화됩니다. 이 백엔드의 풀 스택 CI/CD를 끄려면 이 옵션을 선택 취소하십시오. 풀 스택 CI/CD를 끄면 앱이 풀 전용 모드로 실행됩니다. 빌드 시 Amplify는 백엔드 환경을 수정하지 않고 `aws-exports.js` 파일만 자동으로 생성합니다.
8. 저장(Save)을 선택합니다. Amplify는 다음에 앱을 구축할 때 이러한 변경 사항을 적용합니다.

애플리케이션의 백엔드 빌드

를 AWS Amplify 사용하면에 배포된 데이터, 인증, 스토리지 및 프런트엔드 호스팅을 사용하여 풀스택 애플리케이션을 구축할 수 있습니다 AWS.

AWS Amplify Gen 2는 백엔드를 정의하기 위한 TypeScript 기반 코드 우선 개발자 환경을 도입합니다. Amplify Gen 2를 사용하여 백엔드를 빌드하고 앱에 연결하는 방법을 알아보려면 Amplify 설명서의 백엔드 빌드 및 연결을 참조하세요.

CLI 및 Amplify Studio를 사용하여 Gen 1 앱의 백엔드를 빌드하기 위한 설명서를 찾고 있는 경우 Gen 1 Amplify 설명서의 백엔드 빌드 및 연결을 참조하세요.

주제

- Gen 2 앱의 백엔드 생성
- Gen 1 앱의 백엔드 생성

Gen 2 앱의 백엔드 생성

TypeScript 기반 백엔드를 사용하여 Amplify Gen 2 풀 스택 애플리케이션을 생성하는 단계를 안내하는 자습서는 Amplify 설명서의 시작하기를 참조하세요.

Gen 1 앱의 백엔드 생성

이 자습서에서는 Amplify를 사용하여 풀 스택 CI/CD 워크플로를 설정합니다. Amplify 호스팅에 프런트엔드 앱을 배포합니다. 그런 다음 Amplify Studio를 사용하여 백엔드를 생성합니다. 마지막으로 클라우드 백엔드를 프런트엔드 앱에 연결합니다.

사전 조건

이 자습서를 시작하기 전에 다음 사전 조건을 완료합니다.

에 가입 AWS 계정

> 아직 AWS 고객이 아닌 경우 온라인 지침에 따라 를 생성 AWS 계정해야 합니다. 가입하면 Amplify 및 애플리케이션에 사용할 수 있는 기타 AWS 서비스에 액세스할 수 있습니다.

Git 리포지토리 생성

Amplify는 GitHub, Bitbucket, GitLab 및를 지원합니다 AWS CodeCommit. 애플리케이션을 Git 리포지토리로 푸시합니다.

Amplify 명령줄 인터페이스(CLI) 설치

자세한 지침은 Amplify Framework 설명서의 Amplify CLI 설치를 참조하십시오.

1단계: 프론트엔드 배포

이 예제에 사용하려는 기존 프런트엔드 앱이 git 저장소에 있는 경우, 프런트엔드 앱 배포 안내를 진행하면 됩니다.

이 예제에 사용할 새 프론트엔드 앱을 생성해야 하는 경우 React 앱 생성 설명서의 React 앱 생성 지침을 따를 수 있습니다.

프론트엔드 앱을 배포하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 새 앱을 선택한 다음 오른쪽 상단에서 웹 앱 호스팅을 선택합니다.
3. GitHub, Bitbucket, GitLab 또는 AWS CodeCommit 리포지토리 공급자를 선택한 다음 계속을 선택합니다.
4. Amplify는 git 리포지토리에 대한 액세스를 승인합니다. GitHub 리포지토리의 경우, Amplify는 이제 GitHub 앱 기능을 사용하여 Amplify 액세스를 승인합니다.

   GitHub 앱 설치 및 인증에 대한 자세한 내용은 GitHub 리포지토리에 대한 Amplify 액세스 설정을 참조하십시오.
5. 리포지토리 브랜치 추가 페이지에서 다음을 수행하십시오.

   a. 최근 업데이트된 리포지토리 목록에서 연결할 리포지토리 이름을 선택합니다.

   b. 브랜치 목록에서 연결할 리포지토리 브랜치의 이름을 선택합니다.

   c. 다음을 선택합니다.
6. 보안 설정 구성 페이지에서 다음을 선택합니다.
7. 검토 페이지에서 저장 및 배포를 선택합니다. 배포가 완료되면 `amplifyapp.com` 기본 도메인에서 앱을 볼 수 있습니다.

> **Note**
>
> Amplify 애플리케이션의 보안을 강화하기 위해 amplifyapp.com 도메인은 공개 서픽스 목록(PSL)에 등록되어 있습니다. 보안 강화를 위해 Amplify 애플리케이션의 기본 도메인 이름에 민감한 쿠키를 설정해야 하는 경우, `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도(CSRF)로부터 도메인을 보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

2단계: 백엔드 생성

이제 Amplify 호스팅에 프런트엔드 앱을 배포했으므로 백엔드를 만들 수 있습니다. 다음 지침에 따라 간단한 데이터베이스 및 GraphQL API 엔드포인트가 있는 백엔드를 생성하십시오.

백엔드를 생성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 1단계에서 만든 앱을 선택합니다.
3. 앱 홈페이지에서 백엔드 환경 탭을 선택한 다음 시작하기을 선택합니다. 그러면 기본 스테이징 환경을 위한 설정 프로세스가 시작됩니다.
4. 설정이 완료되면 스튜디오 실행을 선택하여 Amplify Studio의 스테이징 백엔드 환경에 액세스합니다.

Amplify Studio는 백엔드를 생성 및 관리하고 프론트엔드 UI 개발을 가속화하는 시각적 인터페이스입니다. Amplify Studio에 대한 자세한 내용은 Amplify Studio 설명서를 참조하십시오.

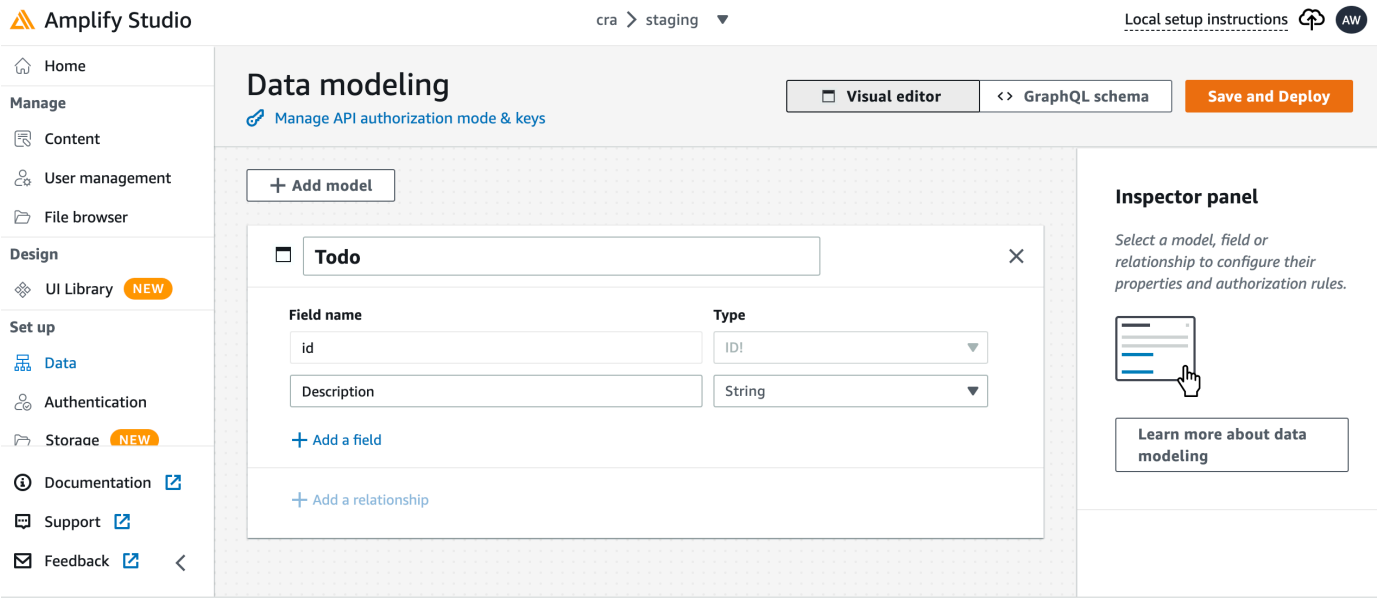
다음 지침에 따라 Amplify Studio 비주얼 백엔드 빌더 인터페이스를 사용하여 간단한 데이터베이스를 생성하십시오.

데이터 모델 생성

1. 앱의 스테이징 환경 홈 페이지에서 데이터 모델 생성을 선택합니다. 그러면 데이터 모델 디자이너가 열립니다.
2. 데이터 모델링 페이지에서 모델 추가를 선택합니다.
3. 제목으로 **Todo**을(를) 입력합니다.
4. 필드 추가를 선택합니다.

5. 모델 이름에 **`Description`**을(를) 입력합니다.

   다음 스크린샷은 디자이너에서 데이터 모델이 어떻게 보일지 보여주는 예입니다.



6. 저장 및 배포를 선택합니다.
7. Amplify Hosting 콘솔로 돌아가면 스테이징 환경 배포가 진행됩니다.

배포 중에 Amplify Studio는 데이터에 액세스하기 위한 an AWS AppSync GraphQL API와 Todo 항목을 호스팅하기 위한 Amazon DynamoDB 테이블을 포함하여 백엔드에 필요한 모든 AWS 리소스를 생성합니다. Amplify는 AWS CloudFormation 를 사용하여 백엔드를 배포하므로 백엔드 정의를 infrastructure-as-code로 저장할 수 있습니다.

3단계: 백엔드를 프런트엔드에 연결

이제 프런트엔드를 배포하고 데이터 모델을 포함하는 클라우드 백엔드를 만들었으므로 프런트엔드를 연결해야 합니다. Amplify CLI를 사용하여 백엔드 정의를 로컬 앱 프로젝트로 가져오려면 다음 지침을 따르십시오.

클라우드 백엔드를 로컬 프런트엔드에 연결하려면

1. 터미널 창을 열고 로컬 프로젝트의 루트 디렉터리로 이동합니다.
2. 터미널 창에서 다음 명령을 실행하여 빨간색 텍스트를 프로젝트의 고유한 앱 ID 및 백엔드 환경 이름으로 대체합니다.

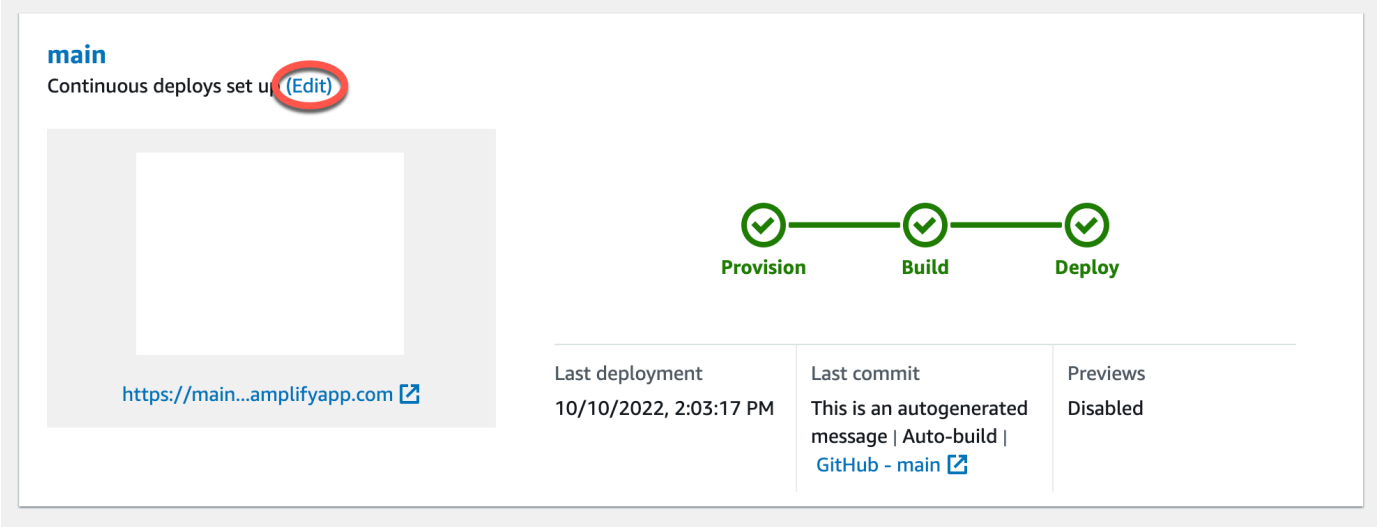
```
amplify pull --appId abcd1234 --envName staging
```

3. 터미널 창의 지침에 따라 프로젝트 설정을 완료하십시오.

이제 지속적 배포 워크플로에 백엔드를 추가하도록 빌드 프로세스를 구성할 수 있습니다. Amplify 호스팅 콘솔에서 프런트엔드 브랜치를 백엔드에 연결하려면 다음 지침을 따르십시오.

프런트엔드 앱 브랜치와 클라우드 백엔드를 연결하려면

1. 앱 홈페이지에서 호스팅 환경 탭을 선택합니다.
2. 기본 브랜치를 찾아 편집을 선택합니다.



3. 대상 백엔드 편집 창의 환경에서 연결할 백엔드의 이름을 선택합니다. 이 예시에서는 2단계에서 만든 스테이징 백엔드를 선택합니다.

   기본적으로 풀 스택 CI/CD가 활성화됩니다. 이 백엔드의 풀 스택 CI/CD를 끄려면 이 옵션을 선택 취소하십시오. 풀 스택 CI/CD를 끄면 앱이 풀 전용 모드로 실행됩니다. 빌드 시 Amplify는 백엔드 환경을 수정하지 않고 `aws-exports.js` 파일만 자동으로 생성합니다.
4. 다음으로 앱 백엔드를 변경하는 데 필요한 권한을 Amplify에 제공하도록 서비스 역할을 설정해야 합니다. 새로운 서비스 역할을 만들거나 기존 역할을 사용할 수 있습니다. 지침은 백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가 단원을 참조하십시오.
5. 서비스 역할을 추가한 후 대상 백엔드 편집 창으로 돌아가서 저장을 선택합니다.
6. 스테이징 백엔드를 프론트엔드 앱의 기본 브랜치에 연결하는 작업을 마치려면 프로젝트의 새 빌드를 수행하십시오.

   다음 중 하나를 수행하십시오.
   - git 리포지토리에서 일부 코드를 푸시하여 Amplify 콘솔에서 빌드를 시작합니다.
   - Amplify 콘솔에서 앱의 빌드 세부 정보 페이지로 이동한 다음 이 버전 재배포를 선택합니다.

다음 단계

기능 브랜치 배포 설정

권장 워크플로에 따라 여러 백엔드 환경에서 기능 브랜치 배포를 설정하십시오.

Amplify Studio에서 프론트엔드 UI 만들기

Studio를 사용하여 바로 사용할 수 있는 UI 구성 요소 세트로 프런트엔드 UI를 구축하고 앱 백엔드에 연결할 수 있습니다. 자세한 내용 및 자습서는 Amplify Framework 설명서에서 Amplify Studio 사용 설명서를 참조하십시오.

고급 배포 기능

이 장에서는 Amplify Hosting 워크플로를 개선하는 고급 배포 기능에 대해 설명합니다. 이러한 기능은 팀이 배포를 보다 효과적으로 관리하고, 코드 품질을 보장하고, 개발 수명 주기 동안 보안을 유지하는 데 도움이 되는 추가 제어 및 기능을 제공합니다.

릴리스되지 않은 기능에 대한 액세스를 제한하기 위해 암호 인증으로 기능 브랜치를 보호하는 방법을 알아봅니다. 코드를 프로덕션 브랜치에 병합하기 전에 고유한 미리 보기 URLs의 변경 사항을 검토하도록 풀 요청에 대한 웹 미리 보기를 활성화합니다. 코드를 프로덕션으로 푸시하기 전에 Cypress 프레임워크를 사용하여 회귀를 포착하는 end-to-end 테스트를 설정합니다. Amplify에 배포 버튼 기능을 더 이상 사용할 수 없지만 Amplify 호스팅을 사용하여 리포지토리에서 직접 애플리케이션을 쉽게 배포할 수 있습니다.

주제

- Amplify 앱의 브랜치에 대한 액세스 제한
- pull 요청를 위한 웹 미리 보기
- Amplify 애플리케이션에 대한 엔드 투 엔드 Cypress 테스트 설정
- Amplify에 배포 버튼을 사용하여 GitHub 프로젝트 공유

Amplify 앱의 브랜치에 대한 액세스 제한

아직 출시되지 않은 기능을 개발 중인 경우, 기능 브랜치를 암호로 보호하여 특정 사용자의 액세스를 제한할 수 있습니다. 브랜치에 액세스 제어가 설정된 경우, 브랜치 URL에 액세스하려고 하면 사용자에게 사용자 이름과 암호를 입력하라는 메시지가 표시됩니다.

개별 브랜치에 적용되는 암호를 설정하거나 연결된 모든 브랜치에 전역적으로 적용되는 암호를 설정할 수 있습니다. 브랜치 및 전역 수준에서 액세스 제어가 활성화되면 브랜치 수준 암호가 전역(애플리케이션) 수준 암호보다 우선합니다.

Amplify는 암호로 보호된 리소스에 액세스하려는 실패한 요청을 제한합니다. 이 동작은 사전 공격 또는 액세스 제어 이후의 데이터를 읽으려는 다른 시도로부터 애플리케이션을 보호합니다.

Amplify 앱의 브랜치에 대한 액세스를 제한하는 암호를 설정하려면 다음 절차를 사용합니다.

기능 브랜치에 암호를 설정하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.

2. 기능 브랜치 암호를 설정하려는 앱을 선택하십시오.
3. 탐색 창에서 호스팅을 선택한 다음 액세스 제어를 선택합니다.
4. 액세스 제어 설정 섹션에서 액세스 관리를 선택합니다.
5. 액세스 제어 관리 페이지에서 다음을 수행합니다.
   - 연결된 모든 브랜치에 적용되는 사용자 이름 및 암호를 설정하려면
     - 모든 브랜치의 액세스 관리를 켭니다. 예를 들어, 기본, 개발 및 기능 브랜치가 연결된 경우모든 브랜치에 동일한 사용자 이름 및 암호를 적용할 수 있습니다.
   - 개별 브랜치에 적용되는 사용자 이름 및 암호를 설정하려면
     a. 모든 브랜치의 액세스 관리를 끕니다.
     b. 관리하려는 브랜치를 찾습니다. 액세스 설정에서 제한됨-암호 필수를 선택합니다.
     c. 사용자 이름에 사용자 이름을 입력합니다.
     d. 암호에는 암호를 입력합니다.
   - 저장을 선택합니다.
6. 서버 측 렌더링(SSR)앱의 액세스 제어를 관리하는 경우, Git 리포지토리에서 새 빌드를 수행하여 앱을 재배포하십시오. 이 단계는 Amplify를 활성화하여 액세스 제어 설정을 적용하는 데 필요합니다.

pull 요청를 위한 웹 미리 보기

웹 미리 보기는 개발 및 품질 보증(QA) 팀이 코드를 프로덕션 또는 통합 브랜치에 병합하기 전에 pull 요청(PR) 의 변경 사항을 미리 볼 수 있는 방법을 제공합니다. pull 요청를 사용하면 리포지토리의 브랜치에 푸시한 변경 내용을 다른 사람에게 알릴 수 있습니다. pull 요청이 열리면 공동 작업자와 잠재적 변경 사항을 논의 및 검토하고 변경 사항을 기본 브랜치에 병합하기 전에 후속 커밋을 추가할 수 있습니다.

웹 미리 보기는 리포지토리에 대한 모든 pull 요청을 기본 사이트에서 사용하는 URL과 완전히 다른 고유한 미리 보기 URL에 배포합니다. Amplify CLI 또는 Amplify Studio를 사용하여 백엔드 환경이 프로비저닝된 앱의 경우 모든 풀 요청(프라이빗 Git 리포지토리만 해당)은 PR이 종료될 때 삭제되는 임시 백엔드를 생성합니다.

앱에 대한 웹 미리 보기가 켜져 있으면 각 PR은 앱당 50개 브랜치의 Amplify 할당량에 포함됩니다. 이 할당량을 초과하지 않으려면 PR을 종료해야 합니다. 할당량에 대한 자세한 내용은 Amplify Hosting 서비스 할당량 섹션을 참조하세요.

> **Note**
>
> 현재를 리포지토리 공급자 AWS CodeCommit 로 사용할 때는 `AWS_PULL_REQUEST_ID` 환경 변수를 사용할 수 없습니다.

웹 미리 보기 보안

보안을 위해 프라이빗 리포지토리가 있는 모든 앱에서 웹 미리 보기를 활성화할 수 있지만 퍼블릭 리포지토리가 있는 모든 앱에서는 활성화할 수 없습니다. Git 저장소가 공용인 경우 IAM 서비스 역할이 필요하지 않은 앱에 대해서만 미리 보기를 설정할 수 있습니다. 예를 들어 백엔드가 있는 앱과 `WEB_COMPUTE` 호스팅 플랫폼에 배포된 앱에는 IAM 서비스 역할이 필요합니다. 따라서 리포지토리가 퍼블릭인 경우 이러한 유형의 앱에 대해 웹 미리 보기를 활성화할 수 없습니다. Amplify는 제3자가 앱의 IAM 역할 권한을 사용하여 실행되는 임의 코드를 제출하지 못하도록 이러한 제한을 적용합니다.

SSR 컴퓨팅 역할을 사용하여 퍼블릭 리포지토리의 애플리케이션에 대해 웹 미리 보기가 활성화된 경우 역할에 액세스할 수 있는 브랜치를 신중하게 관리해야 합니다. 앱 수준 역할은 사용하지 않는 것이 좋습니다. 대신 분기 수준에서 컴퓨팅 역할을 연결해야 합니다. 이렇게 하면 특정 리소스에 액세스해야 하는 브랜치에만 권한을 부여할 수 있습니다. 자세한 정보는 AWS 리소스에 대한 액세스를 허용하는 SSR 컴퓨팅 역할 추가을 참조하십시오.

풀 요청에 대한 웹 미리 보기 활성화

GitHub 리포지토리에 저장된 앱의 경우 웹 미리 보기는 리포지토리 액세스에 Amplify GitHub 앱을 사용합니다. 액세스를 위해 OAuth를 사용하여 이전에 GitHub 리포지토리에서 배포한 기존 Amplify 앱에서 웹 미리 보기를 활성화하려면 먼저 Amplify GitHub 앱을 사용하도록 앱을 마이그레이션해야 합니다. 마이그레이션 지침은 기존 OAuth 앱을 Amplify GitHub 앱으로 마이그레이션하기 단원을 참조하십시오.

pull 요청의 웹 미리 보기를 활성화하려면

1. 호스팅을 선택한 다음 미리 보기를 선택합니다.

   > **Note**
   >
   > 앱이 지속적 배포를 위해 설정되고 Git 리포지토리에 연결된 경우에만 앱 설정 메뉴에 미리 보기가 표시됩니다. 이러한 유형의 배포에 대한 지침은 기존 코드로 시작하기를 참조하십시오.

2. GitHub 리포지토리의 경우에만 다음을 수행하여 계정에 Amplify GitHub 앱을 설치하고 승인하십시오.

   a. 미리 보기를 활성화하려면 GitHub 앱 설치 창에서 GitHub 앱 설치를 선택합니다.

   b. Amplify GitHub 앱을 구성하려는 GitHub 계정을 선택합니다.

   c. 계정에 대한 리포지토리 권한을 구성할 수 있는 페이지가 GitHub.com에서 열립니다.

   d. 다음 중 하나를 수행하십시오.

      - 모든 리포지토리에 설치를 적용하려면 모든 리포지토리를 선택합니다.
      - 선택한 특정 리포지토리로 설치를 제한하려면 리포지토리만 선택을 선택합니다. 선택한 리포지토리에 웹 미리 보기를 활성화하려는 앱의 리포지토리를 포함해야 합니다.

   e. 저장을 선택합니다.

3. 리포지토리의 미리 보기를 활성화한 후 Amplify 콘솔로 돌아가서 특정 분기에 대한 미리 보기를 활성화하십시오. 미리 보기 페이지의 목록에서 브랜치를 선택하고 설정 편집을 선택합니다.

4. 미리 보기 설정 관리 페이지에서 풀 요청 미리 보기를 켭니다. 그 다음 확인을 선택합니다.

5. 풀 스택 애플리케이션의 경우 다음 중 하나를 수행하십시오.

   - 모든 pull 요청에 대해 새 백엔드 환경을 생성을 선택하십시오. 이 옵션을 사용하면 프로덕션에 영향을 주지 않고 변경 사항을 테스트할 수 있습니다.
   - 이 브랜치에 대한 모든 pull 요청이 기존 환경을 가리키도록 설정을 선택합니다.

6. 확인을 선택합니다.

다음에 브랜치에 대한 pull 요청를 제출하면 Amplify는 PR을 빌드하여 미리 보기 URL에 배포합니다. pull 요청이 종료되면 미리 보기 URL이 삭제되고 pull 요청에 연결된 임시 백엔드 환경도 삭제됩니다. GitHub 리포지토리의 경우에만 GitHub 계정의 pull 요청에서 직접 URL 미리 보기에 액세스할 수 있습니다.

하위 도메인을 통한 웹 미리 보기 액세스

Amazon Route 53에서 관리하는 사용자 지정 도메인에 연결된 Amplify 앱의 하위 도메인을 통해 풀 요청 웹 미리 보기에 액세스할 수 있습니다. pull 요청이 종료되면 풀 요청과 관련된 브랜치와 하위 도메인이 자동으로 삭제됩니다. 이는 앱의 패턴 기반 기능 분기 배포를 설정한 후 웹 미리 보기의 기본 동작입니다. 자동 하위 도메인을 설정하는 지침은 Amazon Route 53 사용자 지정 도메인을 위한 자동 하위 도메인 설정 단원을 참조하십시오.

Amplify 애플리케이션에 대한 엔드 투 엔드 Cypress 테스트 설정

Amplify 앱의 테스트 단계에서 엔드 투 엔드(E2E) 테스트를 실행하여 코드를 프로덕션으로 푸시하기 전에 회귀를 포착할 수 있습니다. 테스트 단계는 빌드 사양 YAML에서 구성할 수 있습니다. 현재는 빌드 중에 Cypress 테스트 프레임워크만 실행할 수 있습니다.

Cypress는 브라우저에서 E2E 테스트를 실행할 수 있는 JavaScript 기반 테스트 프레임워크입니다. E2E 테스트를 설정하는 방법을 설명하는 튜토리얼은 풀 스택 CI/CD 배포를 위해 Amplify를 통해 엔드 투 엔드 Cypress 테스트 실행하기 블로그 게시물을 참조하십시오.

기존 Amplify 애플리케이션에 Cypress 테스트 추가

Amplify 콘솔에서 앱의 빌드 설정을 업데이트하여 기존 앱에 Cypress 테스트를 추가할 수 있습니다. 빌드 사양 YAML에는 Amplify에서 빌드를 실행하는 데 사용할 빌드 명령 및 관련 설정 모음이 포함되어 있습니다. 이 `test` 단계를 사용하여 빌드 시 모든 테스트 명령을 실행할 수 있습니다. E2E 테스트의 경우 Amplify Hosting은 테스트용 UI 보고서를 생성할 수 있는 Cypress와의 긴밀한 통합을 제공합니다.

다음 목록에서는 테스트 설정 및 사용 방법에 대해 설명합니다.

테스트 전

Cypress 테스트를 실행하는 데 필요한 종속성을 설치합니다. Amplify Hosting은 mochawesome을 사용하여 보고서를 생성하고 테스트 결과를 확인하며 빌드 중에 로컬 호스트 서버를 설정하기 위해 대기합니다.

테스트

cypress 명령을 실행하여 mochawesome을 사용하는 테스트를 수행합니다.

테스트 후

출력 JSON에서 mochawesome 보고서가 생성됩니다. Yarn을 사용하는 경우 이 명령을 자동 모드에서 실행하여 mochawesome 보고서를 생성해야 한다는 점에 유의합니다. Yarn에는 다음 명령을 사용할 수 있습니다.

```
yarn run --silent mochawesome-merge cypress/report/mochawesome-report/
mochawesome*.json > cypress/report/mochawesome.json
```

artifacts>baseDirectory

테스트가 실행되는 디렉터리입니다.

artifacts>configFilePath

생성된 테스트 보고서 데이터입니다.

artifacts>files

생성된 아티팩트(스크린샷 및 동영상)를 다운로드할 수 있습니다.

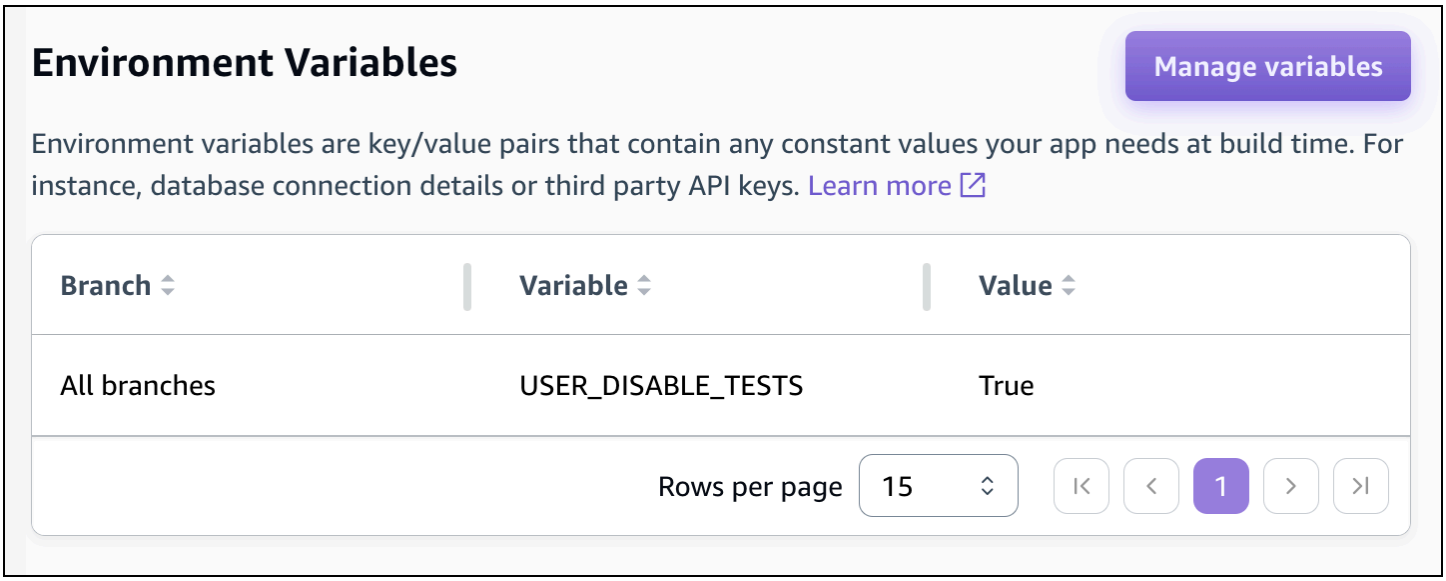
빌드 사양 `amplify.yml` 파일에서 발췌한 다음 예시는 앱에 Cypress 테스트를 추가하는 방법을 설명합니다.

```
test:
  phases:
    preTest:
      commands:
        - npm ci
        - npm install -g pm2
        - npm install -g wait-on
        - npm install mocha mochawesome mochawesome-merge mochawesome-report-generator
        - pm2 start npm -- start
        - wait-on http://localhost:3000
    test:
      commands:
        - 'npx cypress run --reporter mochawesome --reporter-options
 "reportDir=cypress/report/mochawesome-
report,overwrite=false,html=false,json=true,timestamp=mmddyyyy_HHMMss"'
    postTest:
      commands:
        - npx mochawesome-merge cypress/report/mochawesome-report/mochawesome*.json >
 cypress/report/mochawesome.json
        - pm2 kill
  artifacts:
    baseDirectory: cypress
    configFilePath: '**/mochawesome.json'
    files:
      - '**/*.png'
      - '**/*.mp4'
```

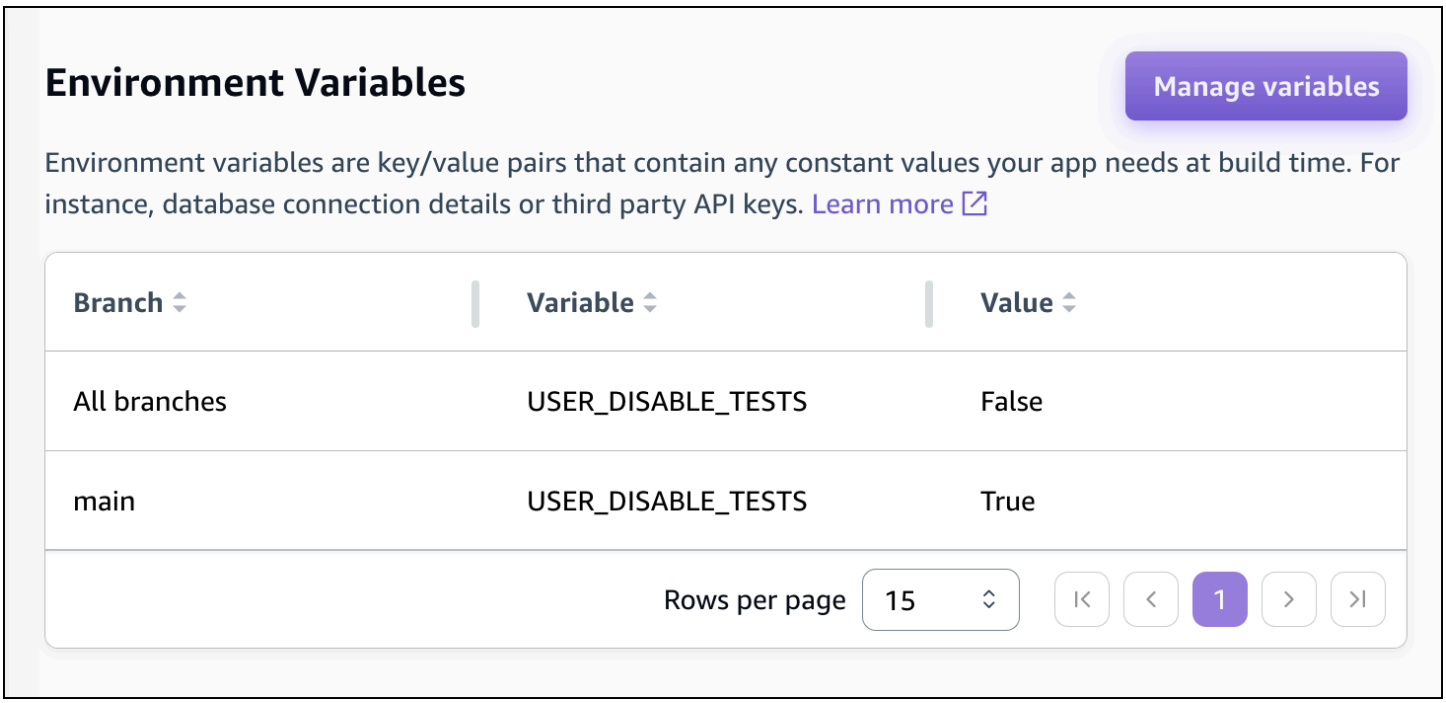
Amplify 애플리케이션 또는 브랜치에 대한 테스트 끄기

테스트 구성을 `amplify.yml` 빌드 설정에 추가한 후에는 모든 브랜치의 모든 빌드에 대해 `test` 단계가 실행됩니다. 테스트 실행을 전역적으로 비활성화하거나 특정 브랜치에 대한 테스트만 실행하려는 경우, 빌드 설정을 수정하는 대신 USER_DISABLE_TESTS 환경 변수를 사용할 수 있습니다.

모든 브랜치에 대한 테스트를 전역적으로 비활성화하려면 모든 브랜치에 대해 값이 `true`인 USER_DISABLE_TESTS 환경 변수를 추가합니다. 다음 스크린샷은 모든 브랜치에 대해 테스트가 비활성화된 Amplify 콘솔의 환경 변수 섹션을 나타냅니다.



특정 브랜치에 대한 테스트를 비활성화하려면 모든 브랜치에 대해 값이 `false`인 USER_DISABLE_TESTS 환경 변수를 추가한 다음, 비활성화하려는 각 브랜치에 대해 값이 `true`인 재정의를 추가합니다. 다음 스크린샷에서는 기본 브랜치에 테스트가 비활성화되고 다른 모든 브랜치에서는 활성화되어 있습니다.



이 변수를 사용하여 테스트를 비활성화하면 빌드 중에 테스트 단계를 완전히 건너뜁니다. 테스트를 다시 활성화하려면 이 값을 `false`로 설정하거나 환경 변수를 삭제합니다.

Amplify에 배포 버튼을 사용하여 GitHub 프로젝트 공유

> **Important**
>
> Amplify Hosting에 배포 버튼을 사용한 원클릭 배포는 더 이상 사용할 수 없습니다. 리포지토리에서 배포하려면 Amplify Hosting에서 새 애플리케이션을 생성합니다. 지침은 Amplify Hosting에 앱 배포 시작하기 섹션을 참조하세요.

Amplify Hosting에 배포 버튼을 사용하면 GitHub 프로젝트를 공개적으로 또는 팀 내에서 공유할 수 있습니다. 다음은 버튼 이미지입니다.

Amplify Hosting에 배포 버튼을 리포지토리 또는 블로그에 추가

버튼을 GitHub README.md 파일, 블로그 게시물, 또는 HTML을 렌더링하는 기타 마크업 페이지에 추가합니다. 버튼은 두 가지 구성 요소를 갖습니다.

1. URL https://oneclick.amplifyapp.com/button.svg에 있는 SVG 이미지
2. GitHub 리포지토리 링크가 포함된 Amplify 콘솔 URL. 리포지토리의 URL(예: https://github.com/username/repository)을 복사하거나 특정 폴더(예: https://github.com/username/repository/tree/branchname/folder)에 딥 링크를 제공할 수 있습니다. Amplify Hosting은 리포지토리의 기본 브랜치를 배포합니다. 앱을 연결한 후에 추가 브랜치를 연결할 수 있습니다.

다음 예제를 사용하여 GitHub Readme.md와 같은 마크다운 파일에 버튼을 추가합니다. https://github.com/username/repository을 리포지토리의 URL로 바꿉니다.

```
[![amplifybutton](https://oneclick.amplifyapp.com/button.svg)](https://
console.aws.amazon.com/amplify/home#/deploy?repo=https://github.com/username/
repository)
```

다음 예제를 사용하여 HTML 문서에 버튼을 추가할 수 있습니다. https://github.com/username/repository을 리포지토리의 URL로 바꿉니다.

```
<a href="https://console.aws.amazon.com/amplify/home#/deploy?repo=https://github.com/
username/repository">
    <img src="https://oneclick.amplifyapp.com/button.svg" alt="Deploy to Amplify
 Hosting">
</a>
```

Amplify 애플리케이션에 대한 리디렉션 및 다시 쓰기 설정

리디렉션을 사용하면 웹 서버가 하나의 URL에서 다른 URL로 경로를 다시 라우팅할 수 있습니다. 리디렉션을 사용하는 일반적인 이유는 다음과 같습니다. URL의 모양을 사용자 지정하고 링크가 끊어지지 않도록 하며 주소를 변경하지 않고 앱이나 사이트의 호스팅 위치를 이동하고 요청된 URL을 웹 앱에 필요한 형식으로 변경합니다.

Amplify가 지원하는 리디렉션 이해

Amplify는 콘솔에서 다음과 같은 리디렉션 유형을 지원합니다.

영구적 리디렉션(301)

301 리디렉션은 웹 주소의 대상 주소를 지속적으로 변경하려는 것입니다. 원본 주소의 검색 엔진 순위 기록이 새 대상 주소에 적용됩니다. 리디렉션은 클라이언트 측에서 발생하므로 리디렉션 후에 브라우저 탐색 모음에 대상 주소가 표시됩니다.

301 리디렉션을 사용하는 일반적인 이유는 다음과 같습니다.

- 페이지 주소가 바뀔 때 링크가 끊어지지 않도록 합니다.
- 사용자가 주소에 흔히 하는 오타를 낼 때 링크가 끊어지지 않도록 합니다.

임시 리디렉션(302)

302 리디렉션은 웹 주소의 대상 주소를 임시로 변경하려는 것입니다. 원본 주소의 검색 엔진 순위 기록이 새 대상 주소에 적용되지 않습니다. 리디렉션은 클라이언트 측에서 발생하므로 리디렉션 후에 브라우저 탐색 모음에 대상 주소가 표시됩니다.

302 리디렉션을 사용하는 일반적인 이유는 다음과 같습니다.

- 원본 주소로 복구하는 동안 우회합니다.
- 사용자 인터페이스의 A/B 비교를 위한 테스트 페이지를 제공합니다.

> **Note**
>
> 앱이 예상치 못한 302 응답을 반환하는 경우, 해당 오류가 발생하는 이유는 앱의 리디렉션 및 사용자 지정 헤더 구성을 변경했기 때문일 수 있습니다. 이 문제를 해결하려면 사용자 지정 헤더가 유효한지 확인한 다음, 앱의 기본 404 다시 쓰기 규칙을 다시 활성화합니다.

다시 쓰기(200)

200 리디렉션(다시 쓰기)은 원본 주소에서 제공된 것처럼 대상 주소의 콘텐츠를 표시하기 위한 것입니다. 검색 엔진 순위 기록이 원본 주소에 계속 적용됩니다. 리디렉션은 서버 측에서 발생하므로 리디렉션 후에 브라우저 탐색 모음에 원래 주소가 표시됩니다. 200 리디렉션을 사용하는 일반적인 이유는 다음과 같습니다.

- 사이트의 주소를 변경하지 않고 전체 사이트를 새 호스팅 위치로 리디렉션합니다.
- 모든 트래픽을 단일 페이지 웹 앱(SPA)의 index.html 페이지로 리디렉션하여 클라이언트 측 라우터 기능으로 처리합니다.

찾을 수 없음(404)

404 리디렉션은 요청이 존재하지 않는 주소를 가리킬 때 발생합니다. 요청된 페이지 대신 404의 대상 주소 페이지가 표시됩니다. 404 리디렉션이 발생하는 일반적인 이유는 다음과 같습니다.

- 사용자가 잘못된 URL을 입력할 때 메시지 링크가 끊어지지 않도록 방지합니다.
- 웹 앱의 존재하지 않는 페이지에 대한 요청이 index.html 페이지를 향하도록 하여 클라이언트 측 라우터 기능으로 처리합니다.

리디렉션 순서 이해

리디렉션은 목록의 맨 위에서 아래로 가며 적용됩니다. 이러한 순서가 의도한 효과를 발휘하는지 확인합니다. 예를 들어 다음과 같은 리디렉션 순서를 통해 /docs/ 아래의 특정 경로에 대한 모든 요청이 /documents/ 아래의 동일한 경로로 리디렉션합니다(단, /docs/specific-filename.html은 /documents/different-filename.html로 리디렉션합니다.

```
/docs/specific-filename.html /documents/different-filename.html 301
/docs/<*> /documents/<*>
```

다음 순서로 리디렉션하면 specific-filename.html의 different-filename.html로 리디렉션이 무시됩니다.

```
/docs/<*> /documents/<*>
/docs/specific-filename.html /documents/different-filename.html 301
```

Amplify가 쿼리 파라미터를 전달하는 방법 이해

쿼리 파라미터를 사용하여 URL 일치에 대한 제어를 강화할 수 있습니다. Amplify는 다음 예외의 경우를 제외하고 모든 쿼리 파라미터를 301 및 302 리디렉션을 위한 대상 경로로 전달합니다.

- 원본 주소에 특정 값으로 설정된 쿼리 문자열이 포함된 경우, Amplify는 쿼리 파라미터를 전달하지 않습니다. 이 경우, 리디렉션은 지정된 쿼리 값을 가진 대상 URL에 대한 요청에만 적용됩니다.
- 일치 규칙의 대상 주소에 쿼리 파라미터가 있는 경우, 쿼리 파라미터는 전달되지 않습니다. 예를 들어 리디렉션의 대상 주소가 `https://example-target.com?q=someParam`인 경우, 쿼리 파라미터는 전달되지 않습니다.

Amplify 콘솔에서 리디렉션 생성 및 편집

Amplify 콘솔에서 애플리케이션에 대한 리디렉션을 생성하고 편집할 수 있습니다. 시작하기 전에 리디렉션의 각 부분에 대한 다음 정보가 필요합니다.

원본 주소

사용자가 요청한 주소입니다.

대상 주소

사용자가 보는 콘텐츠를 실제로 제공하는 주소입니다.

리디렉션 유형

유형에는 영구 리디렉션(301), 임시 리디렉션(302), 다시 쓰기(200) 또는 찾을 수 없음(404)이 포함됩니다.

두 글자로 된 국가 코드(선택 사항)

앱의 사용자 경험을 리전별로 분류하기 위해 포함할 수 있는 값입니다.

Amplify 콘솔에서 리디렉션을 생성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 리디렉션을 생성하려는 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 다시 쓰기 및 리디렉션을 선택합니다.
4. 다시 쓰기 및 리디렉션 페이지에서 리디렉션 관리를 선택합니다.

5. 다시 쓰기 및 리디렉션 JSON 편집기에서 수동으로 리디렉션을 추가하거나 업데이트합니다.

   a. 의 경우 사용자가 요청한 원래 주소를 source지정합니다.

   b. 에서 리디렉션 유형을 status지정합니다.

   c. 에서 사용자에게 콘텐츠를 렌더링하는 대상 주소를 target지정합니다.

   d. (선택 사항)에 2자 국가 코드 조건을 condition입력합니다.

6. 저장을 선택합니다.

리디렉션 및 다시 쓰기 예제 참조

이 섹션에서는 다양한 일반적인 리디렉션 시나리오의 예를 제공합니다. 이러한 예제를 사용하여 Amplify 콘솔 JSON 편집기에서 자체 리디렉션 및 재작성을 생성하기 위한 JSON 구문을 이해할 수 있습니다.

> **Note**
>
> 일치하는 원래 주소 도메인에서는 대소문자를 구분하지 않습니다.

주제

심플 리디렉션 및 다시 쓰기

다음 예제를 사용하여 특정 페이지를 새 주소로 영구적으로 리디렉션할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/original.html	/destination.html	permanent redirect (301)	

JSON 형식

```
[
  {
    "source": "/original.html",
    "status": "301",
    "target": "/destination.html",
    "condition": null
  }
]
```

다음 예제를 사용하여 폴더 아래의 모든 경로를 다른 폴더 아래의 동일한 경로로 리디렉션할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/docs/<*>	/documents/<*>	permanent redirect (301)	

JSON 형식

```
[
  {
    "source": "/docs/<*>",
    "status": "301",
    "target": "/documents/<*>",
    "condition": null
  }
]
```

다음 예제를 사용하여 모든 트래픽을 index.html로 재작성할 수 있습니다. 이 시나리오에서는 다시 쓰기를 통해 원본 주소에 도착했음을 사용자에게 알립니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/<*>	/index.html	rewrite (200)	

JSON 형식

```
[
  {
    "source": "/<*>",
    "status": "200",
    "target": "/index.html",
    "condition": null
  }
]
```

다음 예제를 사용하여 다시 쓰기를 사용하여 사용자에게 표시되는 하위 도메인을 변경할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
https://mydomain.com	https://www.mydomain.com	rewrite (200)	

JSON 형식

```
[
  {
    "source": "https://mydomain.com",
    "status": "200", "target": "https://www.mydomain.com",
    "condition": null
  }
]
```

다음 예제를 사용하여 경로 접두사가 있는 다른 도메인으로 리디렉션할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
https://mydomain.com	https://www.mydomain.com/documents	temporary redirect (302)	

JSON 형식

```
[
  {
    "source": "https://mydomain.com",
    "status": "302",
    "target": "https://www.mydomain.com/documents/",
    "condition": null
  }
]
```

다음 예제를 사용하여 찾을 수 없는 폴더 아래의 경로를 사용자 지정 404 페이지로 리디렉션할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/<*>	/404.html	not found (404)	

JSON 형식

```
[
  {
    "source": "/<*>",
    "status": "404",
    "target": "/404.html",
    "condition": null
  }
]
```

단일 페이지 웹 앱(SPA)에 대한 리디렉션

대부분의 SPA 프레임워크는 HTML5 history.pushState()를 지원하여 서버 요청을 시작하지 않고 브라우저 위치를 변경합니다. 이는 루트(또는 /index.html)에서 여정을 시작하지만 다른 페이지로 직접 이동하는 사용자에게는 적합하지 않습니다.

다음 예제에서는 정규식을 사용하여 정규식에 지정된 특정 파일 확장명을 제외한 모든 파일에 대해 index.html에 대한 200 다시 쓰기를 설정합니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
</^[^.]+$\|\.(?!(css\|gif\|ico\|jpg\|js\|png\|txt\|svg\|woff\|woff2\|ttf\|map\|json\|webp)$)([^.]+$)/>	/index.html	200	

JSON 형식

```
[
  {
    "source": "</^[^.]+$|\.(?!(css|gif|ico|jpg|js|png|txt|svg|woff|woff2|ttf|map|json|
webp)$)([^.]+$)/>",
    "status": "200",
    "target": "/index.html",
    "condition": null
  }
]
```

역방향 프록시 다시 쓰기

다음 예제에서는 다른 위치의 프록시 콘텐츠에 대해 다시 쓰기를 사용하여 사용자에게 도메인이 변경되지 않은 것처럼 표시합니다. HTTPS는 역방향 프록시에 지원되는 유일한 프로토콜입니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/images/<*>	https://images.otherdomain.com/<*>	rewrite (200)	

JSON 형식

```
[
  {
    "source": "/images/<*>",
    "status": "200",
    "target": "https://images.otherdomain.com/<*>",
    "condition": null
  }
]
```

후행 슬래시 및 클린 URL

Hugo와 같은 정적 사이트 생성기는 about.html 대신 about과 같은 깔끔한 URL 구조를 만들기 위해 index.html(/about/index.html)이 있는 페이지 디렉터리를 생성합니다. Amplify는 필요한 경우, 후행 슬래시를 추가하여 자동으로 클린 URL을 생성합니다. 아래 표는 다양한 시나리오를 하이라이트합니다.

브라우저의 사용자 입력	검색 주소창의 URL	제공된 문서
/about	/about	/about.html
/about (when about.html returns 404)	/about/	/about/index.html
/about/	/about/	/about/index.html

자리 표시자

다음 예제를 사용하여 폴더 구조의 경로를 다른 폴더의 일치하는 구조로 리디렉션할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/docs/<year>/<month>/<date>/<itemid>	/documents/<year>/<month>/<date>/<itemid>	permanent redirect (301)	

JSON 형식

```
[
  {
    "source":  "/docs/<year>/<month>/<date>/<itemid>",
    "status": "301",
    "target": "/documents/<year>/<month>/<date>/<itemid>",
               "condition": null
   }
]
```

쿼리 문자열 및 경로 파라미터

> **Warning**
>
> URLs에 비밀, 자격 증명 또는 민감한 데이터를 경로 또는 쿼리 파라미터로 포함하지 마십시오. 이러한 값은 Amplify 애플리케이션의 액세스 로그에서 일반 텍스트로 볼 수 있습니다.

다음 예제를 사용하여 원래 주소의 쿼리 문자열 요소 값과 일치하는 이름을 가진 폴더로 경로를 리디렉션할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/docs?id=<my-blog-id-value	/documents/<my-blog-post-id-value>	permanent redirect (301)	

JSON 형식

```
[
  {
    "source": "/docs?id=<my-blog-id-value>",
    "status": "301",
    "target": "/documents/<my-blog-id-value>",
    "condition": null
  }
]
```

> **Note**
>
> Amplify는 301 및 302 리디렉션에 대한 대상 경로로 모든 쿼리 문자열 파라미터를 전달합니다. 그러나 이 예제에서 볼 수 있듯이 특정 값으로 설정된 쿼리 문자열이 원본 주소에 포함된 경우, Amplify는 쿼리 파라미터를 전달하지 않습니다. 이 경우, 리디렉션은 지정된 쿼리 값 `id`을(를) 가진 대상 주소로 보내는 요청에만 적용됩니다.

다음 예제를 사용하여 폴더 구조의 지정된 수준에서 찾을 수 없는 모든 경로를 지정된 폴더의 index.html로 리디렉션할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
`/documents/<folder>/<child-folder>/<grand-child-folder>`	`/documents/index.html`	`not found (404)`	

JSON 형식

```
[
  {
    "source": "/documents/<x>/<y>/<z>",
    "status": "404",
    "target": "/documents/index.html",
    "condition": null
  }
]
```

리전 기반 리디렉션

다음 예제를 사용하여 리전을 기반으로 요청을 리디렉션할 수 있습니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/documents	/documents/us/	temporary redirect (302)	<US>

JSON 형식

```
[
  {
    "source": "/documents",
    "status": "302",
    "target": "/documents/us/",
    "condition": "<US>"
  }
]
```

리디렉션 및 다시 쓰기에서 와일드카드 표현식 사용

리디렉션 또는 다시 쓰기를 위해 원본 주소에 와일드카드 표현식 <*>를 사용할 수 있습니다. 이 표현식은 원본 주소 끝에 배치해야 하며 고유해야 합니다. Amplify는 둘 이상의 와일드카드 표현식을 포함하는 원본 주소를 무시하거나 다른 배치에 사용합니다.

다음은 와일드카드 표현식이 포함된 유효한 리디렉션의 예입니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/docs/<*>	/documents/<*>	permanent redirect (301)	

다음 두 예제는 와일드카드 표현식을 사용한 잘못된 리디렉션을 보여줍니다.

원본 주소	대상 주소	리디렉션 유형	국가 코드
/docs/<*>/content	/documents/<*>/content	permanent redirect (301)	
/docs/<*>/content/<*>	/documents/<*>/content/<*>	permanent redirect (301)	

Amplify 애플리케이션에서 환경 변수 사용

환경 변수란 Amplify Hosting에서 사용할 수 있도록 애플리케이션 설정에 추가할 수 있는 키-값 페어입니다. 모범 사례로 환경 변수를 사용하여 이러한 애플리케이션 구성 데이터를 노출할 수 있습니다. 추가하는 모든 환경 변수는 불법 액세스를 방지하기 위해 암호화됩니다.

Amplify는 사용자가 생성하는 환경 변수에 대해 다음 제약 조건을 적용합니다.

- Amplify에서는 AWS 접두사가 포함된 환경 변수 이름을 생성할 수 없습니다. 이 접두사는 Amplify 내부 전용으로 예약되어 있습니다.
- 환경 변수의 값은 5,500자를 초과할 수 없습니다.

> **Important**
>
> 환경 변수를 사용하여 암호를 저장하지 않도록 합니다. Gen 2 앱의 경우 Amplify 콘솔에서 비밀 관리 기능을 사용합니다. 자세한 내용은 Amplify 설명서의 비밀 및 환경 변수를 참조하세요. Gen 1 앱의 경우 AWS Systems Manager Parameter Store를 사용하여 생성된 환경 보안 암호에 보안 암호를 저장합니다. 자세한 내용은 환경 비밀 관리 단원을 참조하십시오.

Amplify 환경 변수 참조

다음 환경 변수는 Amplify 콘솔 내에서 기본적으로 액세스할 수 있습니다.

변수 이름	설명	예시 값
_BUILD_TIMEOUT	빌드 제한 시간은 분 단위입니다. 최소값은 5입니다. 최대값은 120입니다.	`30`
_LIVE_UPDATES	도구가 최신 버전으로 업그레이드됩니다.	`[{"name":"Amplify CLI","pkg":"@aws-amplify/cli","type"`

변수 이름	설명	예시 값
		`:"npm","version":"latest"}]`
USER_DISABLE_TESTS	빌드 중에 테스트 단계를 생략합니다. 앱의 모든 브랜치 또는 특정 브랜치에 대한 테스트를 비활성화할 수 있습니다. 이 환경 변수는 빌드 단계에서 테스트를 수행하는 앱에 사용됩니다. 이 변수 설정에 대한 자세한 내용을 알아보려면 Amplify 애플리케이션 또는 브랜치에 대한 테스트 끄기을 참조하십시오.	`true`
AWS_APP_ID	현재 빌드의 앱 ID	`abcd1234`
AWS_BRANCH	현재 빌드의 브랜치 이름	`main`, `develop`, `beta`, `v2.0`
AWS_BRANCH_ARN	현재 빌드의 브랜치 Amazon 리소스 이름(ARN)	`aws:arn:amplify:us-west-2:123456789012:appname/branch/...`
AWS_CLONE_URL	git 리포지토리 콘텐츠를 가져오기 위해 사용된 복제 URL	`git@github.com:<user-name>/<repo-name>.git`
AWS_COMMIT_ID	현재 빌드의 커밋 ID 다시 빌드하기 위한 "HEAD"	`abcd1234`

변수 이름	설명	예시 값
AWS_JOB_ID	현재 빌드의 작업 ID입니다. 여기에는 제로 패딩('0')이 포함되므로 길이가 항상 동일합니다.	`0000000001`
AWS_PULL_REQUEST_ID	풀 요청 웹 미리 보기 빌드의 풀 요청 ID입니다. 를 리포지토리 공급자 AWS CodeCommit 로 사용할 때는이 환경 변수를 사용할 수 없습니다.	`1`
AWS_PULL_REQUEST_SOURCE_BRANCH	Amplify 콘솔에서 애플리케이션 브랜치에 제출되는 풀 요청 미리 보기를 위한 기능 브랜치의 이름입니다.	`featureA`
AWS_PULL_REQUEST_DESTINATION_BRANCH	Amplify 콘솔에서 기능 브랜치 풀 요청이 제출되는 애플리케이션 브랜치의 이름입니다.	`main`
AMPLIFY_AMAZON_CLIENT_ID	Amazon 클라이언트 ID	`123456`
AMPLIFY_AMAZON_CLIENT_SECRET	Amazon 클라이언트 암호	`example123456`
AMPLIFY_FACEBOOK_CLIENT_ID	Facebook 클라이언트 ID	`123456`
AMPLIFY_FACEBOOK_CLIENT_SECRET	Facebook 클라이언트 암호	`example123456`
AMPLIFY_GOOGLE_CLIENT_ID	Google 클라이언트 ID	`123456`

변수 이름	설명	예시 값
AMPLIFY_GOOGLE_CLIENT_SECRET	Google 클라이언트 암호	`example123456`
AMPLIFY_DIFF_DEPLOY	diff 기반 프런트엔드 배포를 활성화 또는 비활성화합니다. 자세한 내용은 diff 기반 프론트엔드 빌드 및 배포 구성 단원을 참조하십시오.	`true`
AMPLIFY_DIFF_DEPLOY_ROOT	리포지토리의 루트를 기준으로 diff 기반 프런트엔드 배포를 비교하는 데 사용할 경로입니다.	`dist`
AMPLIFY_DIFF_BACKEND	diff 기반 백엔드 빌드를 활성화 또는 비활성화합니다. Gen 1 앱에만 적용됩니다. 자세한 내용은 Gen 1 앱을 위한 diff 기반 백엔드 빌드 구성 섹션을 참조하세요.	`true`
AMPLIFY_BACKEND_PULL_ONLY	Amplify는 이 환경 변수를 관리합니다. Gen 1 앱에만 적용됩니다. 자세한 내용은 다른 백엔드를 가리키도록 기존 프론트엔드를 편집합니다. 섹션을 참조하세요.	`true`
AMPLIFY_BACKEND_APP_ID	Amplify는 이 환경 변수를 관리합니다. Gen 1 앱에만 적용됩니다. 자세한 내용은 다른 백엔드를 가리키도록 기존 프론트엔드를 편집합니다. 섹션을 참조하세요.	`abcd1234`

변수 이름	설명	예시 값
AMPLIFY_SKIP_BACKEND_BUILD	빌드 사양에 백엔드 섹션이 없고 백엔드 빌드를 비활성화하려면 이 환경 변수를 `true`으(로) 설정합니다. Gen 1 앱에만 적용됩니다.	`true`
AMPLIFY_ENABLE_DEBUG_OUTPUT	로그에 스택 추적을 인쇄하려면 이 변수를 `true`로 설정합니다. 이는 백엔드 빌드 오류를 디버깅하는 데 유용합니다.	`true`
AMPLIFY_MONOREPO_APP_ROOT	리포지토리의 루트를 기준으로 모노레포 앱의 앱 루트를 지정하는 데 사용할 경로입니다.	`apps/react-app`
AMPLIFY_USERPOOL_ID	인증을 위해 가져온 Amazon Cognito 사용자 풀의 ID	`us-west-2_example`
AMPLIFY_WEBCLIENT_ID	웹 애플리케이션에서 사용할 앱 클라이언트의 ID AMPLIFY_USERPOOL_ID 환경 변수로 지정된 Amazon Cognito 사용자 풀에 액세스할 수 있도록 앱 클라이언트를 구성해야 합니다.	`123456`
AMPLIFY_NATIVECLIENT_ID	네이티브 애플리케이션에서 사용할 앱 클라이언트의 ID AMPLIFY_USERPOOL_ID 환경 변수로 지정된 Amazon Cognito 사용자 풀에 액세스할 수 있도록 앱 클라이언트를 구성해야 합니다.	`123456`

변수 이름	설명	예시 값
AMPLIFY_IDENTITYPOOL_ID	Amazon Cognito 자격 증명 풀의 ID	`example-identitypool-id`
AMPLIFY_PERMISSIONS_BOUNDARY_ARN	Amplify에서 생성한 모든 IAM 역할에 적용되는 권한 경계로 사용할 IAM 정책의 ARN입니다.	`arn:aws:iam::123456789012:policy/example-policy`
AMPLIFY_DESTRUCTIVE_UPDATES	잠재적으로 데이터 손실을 일으킬 수 있는 스키마 작업으로 GraphQL API를 업데이트할 수 있도록 하려면 이 환경 변수를 true로 설정합니다.	`true`

> **Note**
>
> `AMPLIFY_AMAZON_CLIENT_ID` 및 `AMPLIFY_AMAZON_CLIENT_SECRET` 환경 변수는 AWS 액세스 키와 보안 키가 아닌 OAuth 토큰입니다.

프런트엔드 프레임워크 환경 변수

자체 환경 변수를 지원하는 프런트엔드 프레임워크로 앱을 개발하는 경우 이러한 변수는 Amplify 콘솔에 구성하는 환경 변수와 동일하지 않다는 점을 이해하는 것이 중요합니다. 예를 들어 React(접두사: REACT_APP)와 Gatsby(접두사: GATSBY)는 해당 프레임워크가 프런트엔드 프로덕션 빌드에 자동으로 번들링하는 런타임 환경 변수를 만들 수 있습니다. 이러한 환경 변수를 사용하여 값을 저장하는 데 따르는 영향을 이해하려면 사용 중인 프런트엔드 프레임워크 설명서를 참조하십시오.

API 키와 같이 중요한 값을 이러한 프런트엔드 프레임워크 접두사가 붙은 환경 변수 내에 저장하는 것은 모범 사례가 아니므로 권장하지 않습니다.

환경 변수 설정

Amplify 콘솔을 사용하여 애플리케이션의 환경 변수를 설정하려면 다음 지침을 따릅니다.

> **Note**
>
> 앱이 연속 배포되도록 설정하고 git 리포지토리에 연결된 경우에만 Amplify 콘솔의 앱 설정 메뉴에 환경 변수가 표시됩니다. 이러한 유형의 배포에 대한 지침은 기존 코드로 시작하기를 참조하십시오.

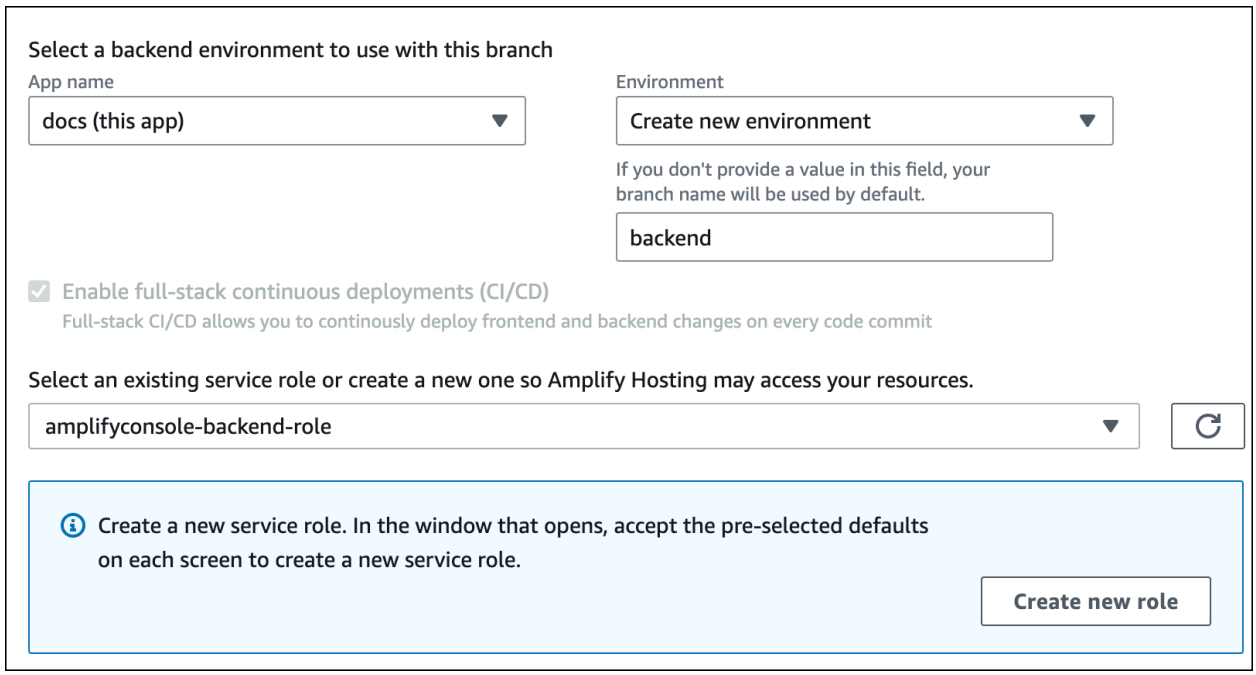
환경 변수를 설정하는 방법

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. Amplify 콘솔에서 호스팅을 선택한 다음 환경 변수를 선택합니다.
3. 환경 변수 페이지에서 변수 관리를 선택합니다.
4. 변수에 키를 입력합니다. 값에 값을 입력합니다. 기본적으로 Amplify는 환경 변수를 모든 브랜치에 적용하므로 새로운 브랜치에 연결할 때 변수를 다시 입력할 필요가 없습니다.
5. (선택 사항) 브랜치에 맞게 환경 변수를 사용자 지정하려면 다음과 같이 브랜치 재정의를 추가합니다.
   a. 작업을 선택한 다음, 변수 재정의 추가를 선택합니다.
   b. 이제 브랜치별 환경 변수 집합을 갖게 되었습니다.
6. 저장을 선택합니다.

소셜 로그인을 위한 인증 파라미터를 사용하여 새 백엔드 환경 생성

브랜치를 앱에 연결하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 브랜치를 앱에 연결하는 절차는 브랜치를 새 앱과 기존 앱 중 어디에 연결하는지에 따라 달라집니다.
   - 브랜치를 새 앱에 연결
     a. 빌드 설정 페이지에서 이 브랜치에 사용할 백엔드 환경 선택 섹션을 찾습니다. 환경에서 새 환경 생성을 선택하고 백엔드 환경의 이름을 입력합니다. 다음 스크린샷은 빌드 설정 페이지의 이 브랜치에 사용할 백엔드 환경 선택 섹션에 백엔드 환경 이름을 **backend**로 입력한 모습입니다.



b. 빌드 설정 페이지에서 고급 설정 섹션을 확장하고 소셜 로그인 키에 대한 환경 변수를 추가합니다. 예를 들어 **AMPLIFY_FACEBOOK_CLIENT_SECRET**은(는) 유효한 환경 변수입니다. 기본적으로 사용할 수 있는 Amplify 시스템 환경 변수 목록은 Amplify 환경 변수 참조의 표를 참조하십시오.

- 브랜치를 기존 앱에 연결

a. 새 브랜치를 기존 앱에 연결하는 경우 브랜치를 연결하기 전에 소셜 로그인 환경 변수를 설정합니다. 탐색 창에서 앱 설정, 환경 변수를 선택합니다.

b. 환경 변수 섹션에서 변수 관리를 선택합니다.

c. 변수 관리 섹션에서 변수 추가를 선택합니다.

d. 변수(키) 에 클라이언트 ID를 입력합니다. 값에 클라이언트 암호를 입력합니다.

e. 저장을 선택합니다.

환경 비밀 관리

Amplify Gen 2 릴리스를 통해 환경 비밀에 대한 워크플로가 간소화되어 Amplify 콘솔에서 비밀 및 환경 변수 관리를 중앙 집중화합니다. Amplify Gen 2 앱의 비밀 설정 및 액세스에 대한 지침은 Amplify 설명서의 비밀 및 환경 변수를 참조하세요.

Gen 1 앱의 환경 비밀은 환경 변수와 비슷하지만 암호화할 수 있는 AWS Systems Manager Parameter Store 키 값 페어입니다. Amplify에 대한 'Apple 프라이빗 키로 로그인'과 같은 일부 값은 암호화되어야 합니다.

AWS Systems Manager 를 사용하여 Amplify Gen 1 애플리케이션의 환경 보안 암호 설정

다음 지침에 따라 AWS Systems Manager 콘솔을 사용하여 Gen 1 Amplify 앱의 환경 암호를 설정합니다.

환경 암호를 설정하려면

1. 에 로그인 AWS Management Console 하고 AWS Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 애플리케이션 관리를 선택한 다음, Parameter Store를 선택합니다.
3. AWS Systems Manager Parameter Store 페이지에서 파라미터 생성을 선택합니다.
4. 파라미터 생성 페이지의 파라미터 세부 정보 섹션에서 다음을 수행합니다.
   a. 이름에 파라미터를 **/amplify/{your_app_id}/{your_backend_environment_name}/{your_parameter_name}** 형식으로 입력합니다.
   b. 유형에서 보안 문자열을 선택합니다.
   c. KMS 키 소스의 경우 내 현재 계정을 선택하여 계정의 기본 키를 사용합니다.
   d. 값 에는 암호화할 암호 값을 입력합니다.
5. 파라미터 생성을 선택합니다.

> **Note**
>
> Amplify는 특정 환경 빌드용 `/amplify/{your_app_id}/{your_backend_environment_name}` 키에만 액세스할 수 있습니다. Amplify가 값을 해독 AWS KMS key 할 수 있도록 기본값을 지정해야 합니다.

Gen 1 애플리케이션의 환경 비밀에 액세스

Gen 1 애플리케이션의 환경 암호는에 JSON 문자열`process.env.secrets`로 저장됩니다.

Amplify 환경 비밀 참조

Systems Manager 파라미터를 `/amplify/{your_app_id}/{your_backend_environment_name}/AMPLIFY_SIWA_CLIENT_ID` 형식으로 지정합니다.

Amplify 콘솔 내에서 기본적으로 액세스할 수 있는 다음 환경 암호를 사용할 수 있습니다.

변수 이름	설명	예시 값
AMPLIFY_SIWA_CLIENT_ID	Apple 클라이언트 ID로 로그인	`com.yourapp.auth`
AMPLIFY_SIWA_TEAM_ID	Apple 팀 ID로 로그인	`ABCD123`
AMPLIFY_SIWA_KEY_ID	Apple 키 ID로 로그인	`ABCD123`
AMPLIFY_SIWA_PRIVATE_KEY	Apple 프라이빗 키로 로그인	-----프라이빗 키 시작----- ****..... -----프라이빗 키 종료-----

Amplify 앱에 대한 사용자 지정 HTTP 헤더 설정

사용자 지정 HTTP 헤더를 사용하면 모든 HTTP 응답에 대해 헤더를 지정할 수 있습니다. 응답 헤더는 디버깅, 보안 및 정보 제공을 목적으로 사용할 수 있습니다. Amplify 콘솔에서 헤더를 지정하거나 앱의 `customHttp.yml` 파일을 다운로드하고 편집한 후 프로젝트의 루트 디렉터리에 저장하여 헤더를 지정할 수 있습니다. 자세한 절차는 사용자 지정 헤더 설정 섹션을 참조하십시오.

이전에는 Amplify 콘솔에서 빌드 사양을 편집하거나 `amplify.yml` 파일을 다운로드하고 업데이트한 다음, 프로젝트의 루트 디렉터리에 저장하는 방식으로 앱에 대한 사용자 지정 HTTP 헤더를 지정했습니다. 이러한 방식으로 지정된 사용자 지정 헤더는 빌드 사양 및 `amplify.yml` 파일 외부로 마이그레이션하는 것이 좋습니다. 지침은 빌드 사양 및 amplify.yml에서 사용자 지정 헤더 마이그레이션 단원을 참조하십시오.

주제

사용자 지정 헤더 YAML 참조

다음 YAML 형식을 사용하여 사용자 지정 헤더를 지정합니다.

```
customHeaders:
  - pattern: '*.json'
    headers:
    - key: 'custom-header-name-1'
      value: 'custom-header-value-1'
    - key: 'custom-header-name-2'
      value: 'custom-header-value-2'
  - pattern: '/path/*'
    headers:
    - key: 'custom-header-name-1'
      value: 'custom-header-value-2'
```

모노레포의 경우, 다음 YAML 형식을 사용합니다.

```
applications:
```

```
  - appRoot: app1
    customHeaders:
    - pattern: '**/*'
      headers:
      - key: 'custom-header-name-1'
        value: 'custom-header-value-1'
  - appRoot: app2
    customHeaders:
    - pattern: '/path/*.json'
      headers:
      - key: 'custom-header-name-2'
        value: 'custom-header-value-2'
```

앱에 사용자 지정 헤더를 추가할 때 다음에 대해 고유한 값을 지정해야 합니다.

패턴

사용자 지정 헤더는 패턴과 일치하는 모든 URL 파일 경로에 적용됩니다.

헤더

파일 패턴과 일치하는 헤더를 정의합니다.

키

사용자 지정 헤더의 이름입니다.

값

사용자 지정 헤더의 값입니다.

HTTP 헤더에 대한 자세한 내용은 Mozilla의 HTTP 헤더 목록을 참조하십시오.

사용자 지정 헤더 설정

Amplify 앱에 사용자 지정 HTTP 헤더를 지정하는 방법은 두 가지가 있습니다. Amplify 콘솔에서 헤더를 지정하거나 앱의 `customHttp.yml` 파일을 다운로드하고 편집한 후 프로젝트의 루트 디렉터리에 저장하여 헤더를 지정할 수 있습니다.

앱에 대한 사용자 지정 헤더를 설정하고 콘솔에 저장하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 사용자 지정 헤더를 설정할 앱을 선택합니다.

3. 탐색 창에서 호스팅을 선택한 다음 사용자 지정 헤더를 선택합니다.
4. 사용자 지정 헤더 페이지에서 편집을 선택합니다.
5. 사용자 지정 헤더 편집 창에서 사용자 지정 헤더 YAML 형식을 사용하여 사용자 지정 헤더의 정보를 입력합니다.

   a. `pattern`에 일치시킬 패턴을 입력합니다.

   b. `key`에 사용자 지정 헤더의 이름을 입력합니다.

   c. `value`에 사용자 지정 헤더의 값을 입력합니다.
6. 저장을 선택합니다.
7. 앱을 재배포하여 새 사용자 지정 헤더를 적용합니다.
   - CI/CD 앱의 경우, 배포할 브랜치로 이동한 다음 이 버전 재배포를 선택합니다. Git 리포지토리에서 새 빌드를 수행할 수도 있습니다.
   - 수동 배포 앱의 경우, Amplify 콘솔에서 앱을 다시 배포합니다.

앱에 대한 사용자 지정 헤더를 설정하고 리포지토리의 루트에 저장하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 사용자 지정 헤더를 설정할 앱을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 사용자 지정 헤더를 선택합니다.
4. 사용자 지정 헤더 페이지에서 YML 다운로드를 선택합니다.
5. 다운로드한 `customHttp.yml` 파일을 선택한 코드 편집기에서 열고 사용자 지정 헤더 YAML 형식을 사용하여 사용자 지정 헤더에 대한 정보를 입력합니다.

   a. `pattern`에 일치시킬 패턴을 입력합니다.

   b. `key`에 사용자 지정 헤더의 이름을 입력합니다.

   c. `value`에 사용자 지정 헤더의 값을 입력합니다.
6. 편집된 `customHttp.yml` 파일을 프로젝트의 루트 디렉터리에 저장합니다. 모노레포로 작업하는 경우, 리포지토리의 루트에 `customHttp.yml` 파일을 저장합니다.
7. 앱을 재배포하여 새 사용자 지정 헤더를 적용합니다.
   - CI/CD 앱의 경우, 새 `customHttp.yml` 파일이 포함된 Git 리포지토리에서 새 빌드를 수행합니다.
   - 수동 배포 앱의 경우, Amplify 콘솔에서 앱을 다시 배포하고 업로드하는 아티팩트와 함께 새 `customHttp.yml` 파일을 포함합니다.

> **Note**
>
> `customHttp.yml` 파일에 설정되고 앱의 루트 디렉터리에 배포된 사용자 지정 헤더는 Amplify 콘솔의 사용자 지정 헤더 섹션에 정의된 사용자 지정 헤더를 재정의합니다.

보안 사용자 지정 헤더 예제

사용자 지정 보안 헤더는 HTTPS 실행, XSS 공격 예방 및 클릭재킹으로부터 브라우저 보호를 활성화합니다. 다음 YAML 구문을 사용하여 앱에 사용자 지정 보안 헤더를 적용할 수 있습니다.

```
customHeaders:
  - pattern: '**'
    headers:
      - key: 'Strict-Transport-Security'
        value: 'max-age=31536000; includeSubDomains'
      - key: 'X-Frame-Options'
        value: 'SAMEORIGIN'
      - key: 'X-XSS-Protection'
        value: '1; mode=block'
      - key: 'X-Content-Type-Options'
        value: 'nosniff'
      - key: 'Content-Security-Policy'
        value: "default-src 'self'"
```

Cache-Control 사용자 지정 헤더 설정

Amplify로 호스팅되는 앱은 사용자가 정의하는 사용자 지정 헤더로 재정의하지 않는 한 오리진에서 전송되는 `Cache-Control` 헤더를 그대로 사용합니다. Amplify는 `200 OK` 상태 코드를 반환하는 성공적인 응답에 대해서만 Cache-Control 사용자 지정 헤더를 적용합니다. 이렇게 하면 오류 응답이 캐시되어 동일한 요청을 하는 다른 사용자에게 제공되는 것을 방지할 수 있습니다.

앱의 성능 및 배포 가용성을 더 잘 제어하도록 `s-maxage` 디렉티브를 수동으로 조정할 수 있습니다. 예를 들어 콘텐츠가 엣지에 캐시되는 시간을 늘리려면 `s-maxage`을(를) 기본값인 600초(10분)보다 긴 값으로 업데이트하여 TTL(Time to Live)을 수동으로 늘릴 수 있습니다.

`s-maxage`에 사용자 지정 값을 지정하려면 다음 YAML 형식을 사용합니다. 이 예제에서는 관련 콘텐츠를 3,600초(1시간) 동안 엣지에 캐시된 상태로 유지합니다.

```
customHeaders:
```

```
  - pattern: '/img/*'
    headers:
      - key: 'Cache-Control'
        value: 's-maxage=3600'
```

헤더로 애플리케이션 성능을 제어하는 방법에 대한 자세한 내용은 Cache-Control 헤더를 사용하여 앱 성능 향상을 참조하십시오.

빌드 사양 및 amplify.yml에서 사용자 지정 헤더 마이그레이션

이전에는 Amplify 콘솔에서 빌드 사양을 편집하거나 `amplify.yml` 파일을 다운로드하고 업데이트한 다음, 프로젝트의 루트 디렉터리에 저장하여 앱에 대한 사용자 지정 HTTP 헤더를 지정했습니다. 사용자 지정 헤더를 빌드 사양 및 `amplify.yml` 파일 외부로 마이그레이션하는 것이 좋습니다.

Amplify 콘솔의 사용자 지정 헤더 섹션에서 또는 `customHttp.yml` 파일을 다운로드하고 편집하여 사용자 지정 헤더를 지정합니다.

Amplify 콘솔에 저장된 사용자 지정 헤더를 마이그레이션하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 사용자 지정 헤더 마이그레이션을 수행할 앱을 선택합니다.
3. 탐색 창에서 호스팅, 빌드 설정을 선택합니다. 앱 빌드 사양 섹션에서 앱의 buildspec을 검토할 수 있습니다.
4. 다운로드를 선택하여 현재 buildspec의 사본을 저장합니다. 설정을 복구해야 하는 경우, 나중에 이 사본을 참조할 수 있습니다.
5. 다운로드가 완료되면 편집을 선택합니다.
6. 나중에 9단계에서 사용하게 되므로 파일의 사용자 지정 헤더 정보를 기록해 둡니다. 편집 창에서 파일의 사용자 지정 헤더를 삭제하고 저장을 선택합니다.
7. 탐색 창에서 호스팅, 사용자 지정 헤더를 선택합니다.
8. 사용자 지정 헤더 페이지에서 편집을 선택합니다.
9. 6단계에서 삭제한 사용자 지정 헤더의 정보를 사용자 지정 헤더 편집 창에 입력합니다.
10. 저장(Save)을 선택합니다.
11. 새 사용자 지정 헤더를 적용할 브랜치를 재배포합니다.

사용자 지정 헤더를 amplify.yml에서 customHttp.yml로 마이그레이션하려면

1. 앱의 루트 디렉터리에 현재 배포된 `amplify.yml` 파일로 이동합니다.
2. 원하는 코드 편집기에서 `amplify.yml` 파일을 엽니다.
3. 나중에 8단계에서 사용하게 되므로 파일의 사용자 지정 헤더 정보를 기록해 둡니다. 파일의 사용자 지정 헤더를 삭제합니다. 파일을 저장하고 닫습니다.
4. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
5. 사용자 지정 헤더를 설정할 앱을 선택합니다.
6. 탐색 창에서 호스팅, 사용자 지정 헤더를 선택합니다.
7. 사용자 지정 헤더 페이지에서 다운로드를 선택합니다.
8. 다운로드한 `customHttp.yml` 파일을 원하는 코드 편집기에서 열고 3단계의 `amplify.yml`에서 삭제한 사용자 지정 헤더의 정보를 입력합니다.
9. 편집된 `customHttp.yml` 파일을 프로젝트의 루트 디렉터리에 저장합니다. 모노레포로 작업하는 경우, 리포지토리의 루트에 파일을 저장합니다.
10. 앱을 재배포하여 새 사용자 지정 헤더를 적용합니다.
    - CI/CD 앱의 경우, 새 `customHttp.yml` 파일이 포함된 Git 리포지토리에서 새 빌드를 수행합니다.
    - 수동 배포 앱의 경우, Amplify 콘솔에서 앱을 다시 배포하고 업로드하는 아티팩트와 함께 새 `customHttp.yml` 파일을 포함합니다.

> **Note**
>
> `customHttp.yml` 파일에 설정되고 앱의 루트 디렉터리에 배포된 사용자 지정 헤더는 Amplify 콘솔의 사용자 지정 헤더 섹션에 정의된 사용자 지정 헤더를 재정의합니다.

모노 리포지토리 사용자 지정 헤더 요구 사항

모노레포 앱에 대한 사용자 지정 헤더를 지정할 때는 다음 설정 요구 사항에 유의합니다.

- 모노레포에는 특정 YAML 형식이 있습니다. 사용자 지정 헤더 YAML 참조 섹션에서 올바른 구문을 참조하십시오.

- Amplify 콘솔의 사용자 지정 헤더 섹션을 사용하여 모노 리포지토리의 애플리케이션에 대한 사용자 지정 헤더를 지정할 수 있습니다. 새 사용자 지정 헤더를 적용하려면 애플리케이션을 재배포해야 합니다.
- 콘솔을 사용하는 대신 모노레포 앱에 대한 사용자 지정 헤더를 `customHttp.yml` 파일에 지정할 수 있습니다. `customHttp.yml` 파일을 리포지토리의 루트에 저장한 다음, 애플리케이션을 재배포하여 새 사용자 지정 헤더를 적용해야 합니다. `customHttp.yml` 파일에 지정된 사용자 지정 헤더는 Amplify 콘솔의 사용자 지정 헤더 섹션을 사용하여 지정된 모든 사용자 지정 헤더를 재정의합니다.

앱의 캐시 구성 관리

Amplify는 Amazon CloudFront를 사용하여 호스팅된 애플리케이션의 캐싱 구성을 관리합니다. 최상의 성능을 위해 각 앱에 캐시 구성이 적용됩니다.

2024년 8월 13일, Amplify는 애플리케이션의 캐싱 효율성을 개선했습니다. 자세한 내용은 AWS Amplify 호스팅을 통한 앱 성능 향상을 위한 CDN 캐싱 개선 사항을 참조하세요.

다음 표에는 캐싱 개선 사항 릴리스 전후의 특정 캐싱 동작에 대한 Amplify 지원이 요약되어 있습니다.

캐싱 동작	이전 지원	캐싱 개선 이후
Amplify 콘솔 또는 `customHeaders.yaml` 파일에서 앱의 사용자 지정 헤더를 추가할 수 있습니다. 재정의할 수 있는 헤더 중 하나는 `Cache-Control` 입니다. 자세한 내용은 Amplify 앱에 대한 사용자 지정 HTTP 헤더 설정 단원을 참조하십시오.	예	예
Amplify는 `customHeaders.yaml` 파일에서 정의한 `Cache-Control` 헤더를 준수하며 이 헤더는 Amplify의 기본 캐시 설정보다 우선합니다.	예	예
Amplify는 애플리케이션의 프레임워크 내에서 동적 경로(예: Next.js SSR 경로)에 대해 설정된 `Cache-Control` 헤더를 준수합니다. 앱의 `customHeaders.yaml` 파일에 `Cache-Control` 헤더가 설정된 경우 `next.conf`	예	예

캐싱 동작	이전 지원	캐싱 개선 이후
`ig.js` 파일의 설정보다 우선합니다.		
각 새 CI/CD 앱 배포는 캐시를 지웁니다.	예	예
앱에서 성능 모드를 켤 수 있습니다.	예	아니요 Amplify 콘솔에서는 더 이상 성능 모드 설정을 사용할 수 없습니다. 그러나 `s-maxage` 지시문을 설정하는 `Cache-Control` 헤더를 생성할 수 있습니다. 지침은 Cache-Control 헤더를 사용하여 앱 성능 향상 섹션을 참조하세요.

다음 표에는 특정 캐시 설정의 기본값에 대한 변경 사항이 나열되어 있습니다.

캐시 설정	이전 기본값	캐싱 개선 이후 기본값
정적 자산의 캐시 기간	2초	1년
역방향 프록시 응답의 캐시 기간	2초	0초(캐싱 없음)
최대 TTL(Time To Live)	10분	1년

Amplify가 애플리케이션에 적용할 캐싱 구성을 결정하는 방법과 캐시 키 구성 관리에 대한 지침은 다음 항목을 참조하세요.

주제

- Amplify가 앱에 캐시 구성을 적용하는 방법
- 캐시 키 쿠키 관리
- Cache-Control 헤더를 사용하여 앱 성능 향상

Amplify가 앱에 캐시 구성을 적용하는 방법

앱의 캐싱을 관리하기 위해 Amplify는 앱의 플랫폼 유형 및 다시 쓰기 규칙을 검사하여 제공되는 콘텐츠의 유형을 결정합니다. `Compute` 앱의 경우 Amplify는 배포 매니페스트의 라우팅 규칙도 검사합니다.

> **Note**
>
> 앱의 플랫폼 유형은 배포 중에 Amplify Hosting에서 설정합니다. SSG(정적) 앱의 플랫폼 유형은 `WEB`으로 설정됩니다. SSR 앱(Next.js 12 이상)의 플랫폼 유형은 `WEB_COMPUTE`로 설정됩니다.

Amplify는 다음 네 가지 유형의 콘텐츠를 식별하고 지정된 관리형 캐시 정책을 적용합니다.

정적

`WEB` 플랫폼을 사용하는 앱 또는 `WEB_COMPUTE` 앱의 정적 경로에서 제공되는 콘텐츠입니다.

이 콘텐츠는 Amplify-StaticContent 캐시 정책을 사용합니다.

이미지 최적화

`WEB_COMPUTE` 앱의 `ImageOptimization` 경로에서 제공되는 이미지입니다.

이 콘텐츠는 Amplify-ImageOptimization 캐시 정책을 사용합니다.

컴퓨팅

`WEB_COMPUTE` 앱의 `Compute` 경로에서 제공되는 콘텐츠입니다. 여기에는 모든 서버 측 렌더링(SSR) 콘텐츠가 포함됩니다.

이 콘텐츠는 Amplify App에 설정된 `cacheConfig.type`의 값에 따라 Amplify-Default 또는 Amplify-DefaultNoCookies 캐시 정책을 사용합니다.

역방향 프록시

역방향 프록시 다시 쓰기 사용자 지정 규칙과 일치하는 경로에서 제공되는 콘텐츠입니다. 이 사용자 지정 규칙을 생성하는 방법에 대한 자세한 내용은 리디렉션 사용 장의 역방향 프록시 다시 쓰기 섹션을 참조하세요.

이 콘텐츠는 Amplify App에 설정된 `cacheConfig.type`의 값에 따라 Amplify-Default 또는 Amplify-DefaultNoCookies 캐시 정책을 사용합니다.

Amplify의 관리형 캐시 정책 이해

Amplify는 다음과 같은 사전 정의된 관리형 캐시 정책을 사용하여 호스팅된 애플리케이션의 기본 캐시 구성을 최적화합니다.

- Amplify-Default
- Amplify-DefaultNoCookies
- Amplify-ImageOptimization
- Amplify-StaticContent

Amplify-Default 관리형 캐시 정책 설정

CloudFront 콘솔에서 이 정책 보기

이 정책은 AWS Amplify 웹 앱인 오리진과 함께 사용하도록 설계되었습니다.

이 정책에는 다음 설정이 포함되어 있습니다.

- 최소 TTL: 0초
- 최대 TTL: 31536000초(1년)
- 기본 TTL: 0초
- 캐시 키에 포함된 헤더:
  - `Authorization`
  - `Accept`
  - `CloudFront-Viewer-Country`
  - `Host`
- 캐시 키에 포함된 쿠키: 모든 쿠키가 포함됩니다.
- 캐시 키에 포함된 쿼리 문자열: 모든 쿼리 문자열이 포함됩니다.
- 압축된 객체 캐시 설정: Gzip 및 Brotli에 대해 활성화됩니다.

Amplify-DefaultNoCookies 관리형 캐시 정책 설정

CloudFront 콘솔에서 이 정책 보기

이 정책은 AWS Amplify 웹 앱인 오리진과 함께 사용하도록 설계되었습니다.

이 정책에는 다음 설정이 포함되어 있습니다.

- 최소 TTL: 0초
- 최대 TTL: 31536000초(1년)
- 기본 TTL: 0초
- 캐시 키에 포함된 헤더:
  - `Authorization`
  - `Accept`
  - `CloudFront-Viewer-Country`
  - `Host`
- 캐시 키에 포함된 쿠키: 어떤 쿠키도 포함되지 않습니다.
- 캐시 키에 포함된 쿼리 문자열: 모든 쿼리 문자열이 포함됩니다.
- 압축된 객체 캐시 설정: Gzip 및 Brotli에 대해 활성화됩니다.

Amplify-ImageOptimization 관리형 캐시 정책 설정

CloudFront 콘솔에서 이 정책 보기

이 정책은 AWS Amplify 웹 앱인 오리진과 함께 사용하도록 설계되었습니다.

이 정책에는 다음 설정이 포함되어 있습니다.

- 최소 TTL: 0초
- 최대 TTL: 31536000초(1년)
- 기본 TTL: 0초
- 캐시 키에 포함된 헤더:
  - `Authorization`
  - `Accept`
  - `Host`
- 캐시 키에 포함된 쿠키: 어떤 쿠키도 포함되지 않습니다.
- 캐시 키에 포함된 쿼리 문자열: 모든 쿼리 문자열이 포함됩니다.
- 압축된 객체 캐시 설정: Gzip 및 Brotli에 대해 활성화됩니다.

Amplify-StaticContent 관리형 캐시 정책 설정

CloudFront 콘솔에서 이 정책 보기

이 정책은 AWS Amplify 웹 앱인 오리진과 함께 사용하도록 설계되었습니다.

이 정책에는 다음 설정이 포함되어 있습니다.

- 최소 TTL: 0초
- 최대 TTL: 31536000초(1년)
- 기본 TTL: 0초
- 캐시 키에 포함된 헤더:
  - `Authorization`
  - `Host`
- 캐시 키에 포함된 쿠키: 어떤 쿠키도 포함되지 않습니다.
- 캐시 키에 포함된 쿼리 문자열: 어떤 쿼리 문자열도 포함되지 않습니다.
- 압축된 객체 캐시 설정: Gzip 및 Brotli에 대해 활성화됩니다.

캐시 키 쿠키 관리

Amplify에 앱을 배포할 때 캐시 키에 쿠키를 포함할지 아니면 제외할지 선택할 수 있습니다. Amplify 콘솔에서 이 설정은 사용자 지정 헤더 및 캐시 페이지에서 캐시 키 설정 토글을 사용하여 지정됩니다. 지침은 캐시 키에서 쿠키 포함 또는 제외 섹션을 참조하세요.

캐시 키에 쿠키 포함

이 설정을 사용하면 Amplify는 제공되는 콘텐츠 유형에 따라 앱에 대한 최적의 캐시 구성을 자동으로 선택합니다. 이 캐시 구성 유형은 명시적으로 선택해야 합니다.

SDKs 또는를 사용하는 경우 AWS CLI이 설정은 `CreateApp` 또는 `UpdateApp` APIs를 `AMPLIFY_MANAGED` 사용하여를 `cacheConfig.type`로 설정하는 것에 해당합니다.

캐시 키에서 쿠키 제외

기본 캐시 구성입니다. 이 캐시 구성은 캐시 키에서 모든 쿠키를 제외한다는 점을 제외하면 `AMPLIFY_MANAGED` 구성과 유사합니다.

캐시 키에서 쿠키를 제외하도록 선택하면 캐시 성능이 향상될 수 있습니다. 그러나 이 캐시 구성을 선택하기 전에 앱이 쿠키를 사용하여 동적 콘텐츠를 제공하는지 여부를 고려하는 것이 중요합니다.

SDKs 또는를 사용하는 경우 AWS CLI이 설정은 `CreateApp` 또는 `UpdateApp` APIs를 사용하여를 `cacheConfig.type`AMPLIFY_MANAGED_NO_COOKIES로 설정하는 것에 해당합니다.

캐시 키에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 캐시 키 이해를 참조하세요.

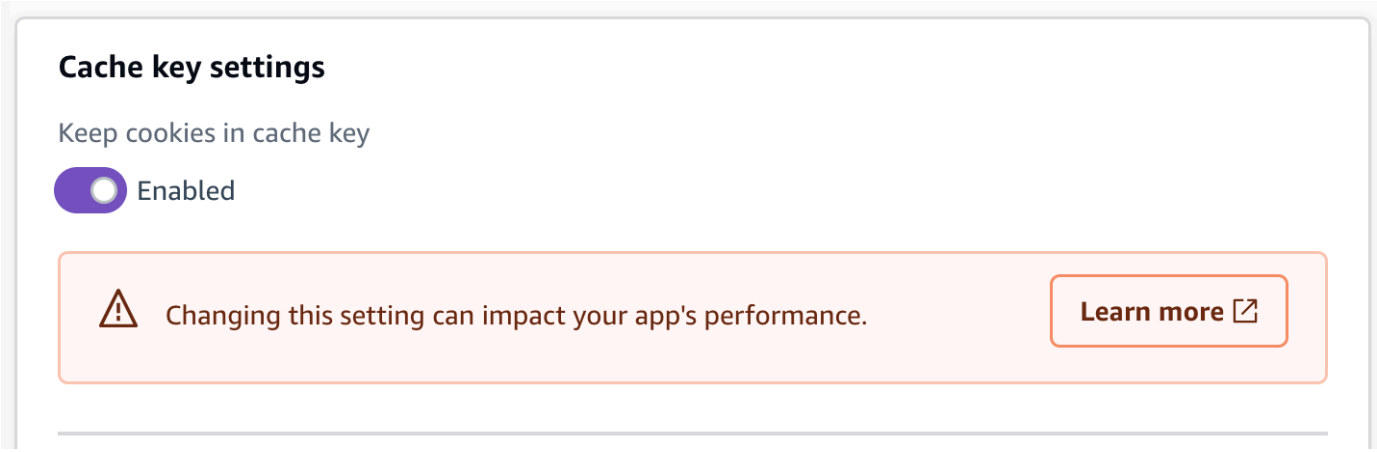
캐시 키에서 쿠키 포함 또는 제외

Amplify 콘솔, SDK 또는 AWS CLI에서 앱에 대한 캐시 키 쿠키 구성을 설정할 수 있습니다.

Amplify 콘솔을 사용하여 새 앱을 배포할 때 캐시 키에서 쿠키를 포함할지 아니면 제외할지 지정하려면 다음 절차를 사용합니다.

Amplify에 앱을 배포할 때 캐시 키 쿠키 구성을 설정하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 새 앱 생성을 선택합니다.
3. Amplify로 빌드 시작 페이지에서 Git 리포지토리 공급자를 선택하고 다음을 선택합니다.
4. 리포지토리 브랜치 추가 페이지에서 다음을 수행합니다.
   a. 연결할 리포지토리의 이름을 선택합니다.
   b. 연결할 리포지토리 브랜치의 이름을 선택합니다.
   c. 다음을 선택합니다.
5. 앱에 IAM 서비스 역할이 필요한 경우 Amplify Hosting 컴퓨팅이 자동으로 서비스 역할을 생성하도록 허용하거나 사용자가 생성한 역할을 지정할 수 있습니다.
   - Amplify가 자동으로 역할을 생성하여 앱에 연결할 수 있도록 하려면
     - 새 서비스 역할 생성 및 사용을 선택합니다.
   - 이전에 생성한 서비스 역할을 연결하려면
     a. 기존 서비스 역할 사용을 선택합니다.
     b. 목록에서 사용할 역할을 선택합니다.
6. 고급 설정을 선택한 다음 캐시 키 설정 섹션을 찾습니다.
7. 쿠키를 캐시 키에 보관 또는 캐시 키에서 쿠키 제거를 선택합니다. 다음 스크린샷은 콘솔의 캐시 키 설정 토글을 보여줍니다.



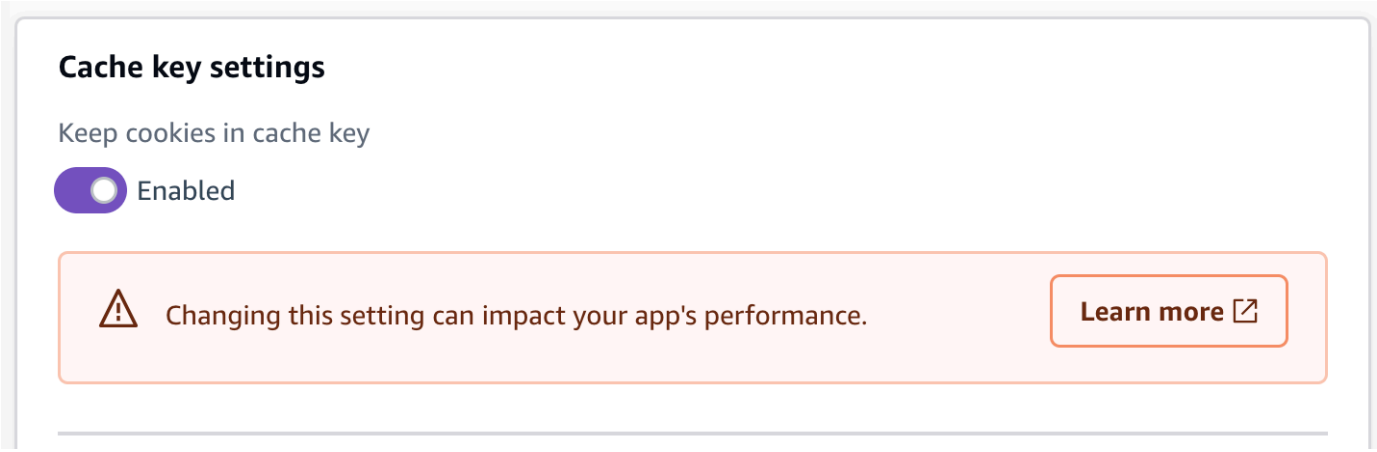
8. 다음을 선택합니다.
9. 검토 페이지에서 저장 및 배포를 선택합니다.

앱의 캐시 키 쿠키 구성 변경

Amplify에 이미 배포된 앱의 캐시 키 쿠키 구성을 변경할 수 있습니다. Amplify 콘솔을 사용하여 앱의 캐시 키에서 쿠키를 포함할지 아니면 제외할지 여부를 변경하려면 다음 절차를 사용합니다.

배포된 앱의 캐시 키 쿠키 구성을 변경하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 업데이트하려는 애플리케이션을 선택합니다.
3. 탐색 창에서 호스팅을 선택한 다음 사용자 지정 헤더 및 캐시를 선택합니다.
4. 사용자 지정 헤더 및 캐시 페이지에서 캐시 키 설정 섹션을 찾아 편집을 선택합니다.
5. 쿠키를 캐시 키에 보관 또는 캐시 키에서 쿠키 제거를 선택합니다. 다음 스크린샷은 콘솔의 캐시 키 설정 토글을 보여줍니다.



6. 저장을 선택합니다.

Cache-Control 헤더를 사용하여 앱 성능 향상

Amplify의 기본 호스팅 아키텍처는 호스팅 성능과 배포 가용성 간의 균형을 최적화합니다. 대부분의 고객은 기본 아키텍처를 사용하는 것이 좋습니다.

앱의 성능을 더 세밀하게 제어해야 하는 경우 콘텐츠 전송 네트워크(CDN) 엣지에서 콘텐츠를 더 긴 간격 동안 캐시하여 호스팅 성능을 최적화하도록 HTTP `Cache-Control` 헤더를 수동으로 설정할 수 있습니다.

HTTP `Cache-Control` 헤더 `max-age` 및 `s-maxage` 명령은 앱의 콘텐츠 캐싱 기간에 영향을 줍니다. `max-age` 명령을 사용하면 오리진 서버에서 콘텐츠를 새로 고치기 전에 캐시에 유지하려는 시간(초)을 브라우저에 알려 줍니다. `s-maxage` 명령을 사용하면 `max-age`을(를) 재정의하고 오리진 서버에서 콘텐츠를 새로 고치기 전에 CDN 엣지에 보관하려는 시간(초)을 지정할 수 있습니다.

Amplify로 호스팅되는 앱은 사용자가 정의하는 사용자 지정 헤더로 재정의하지 않는 한 오리진에서 전송되는 `Cache-Control` 헤더를 그대로 사용합니다. Amplify는 `200 OK` 상태 코드를 반환하는 성공적인 응답에 대해서만 `Cache-Control` 사용자 지정 헤더를 적용합니다. 이렇게 하면 오류 응답이 캐시되어 동일한 요청을 하는 다른 사용자에게 제공되는 것을 방지할 수 있습니다.

앱의 성능 및 배포 가용성을 더 잘 제어하도록 `s-maxage` 디렉티브를 수동으로 조정할 수 있습니다. 예를 들어, 콘텐츠가 엣지에 캐시되는 시간을 변경하려면 `s-maxage`를 기본값인 31536000초(1년) 이외의 값으로 업데이트하여 TTL(Time to Live)을 수동으로 설정할 수 있습니다.

Amplify 콘솔의 사용자 지정 헤더 섹션에서 앱의 사용자 지정 헤더를 정의할 수 있습니다. YAML 형식의 예는 Cache-Control 사용자 지정 헤더 설정 단원을 참조하십시오.

CDN 엣지에서 24시간 동안 콘텐츠를 캐시하도록 `s-maxage` 지시문을 설정하려면 다음 절차를 사용합니다.

사용자 지정 Cache-Control 헤더를 설정하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 사용자 지정 헤더를 설정할 앱을 선택합니다.
3. 탐색 창에서 호스팅, 사용자 지정 헤더를 선택합니다.
4. 사용자 지정 헤더 페이지에서 편집을 선택합니다.
5. 사용자 지정 헤더 편집 창에서 사용자 지정 헤더의 정보를 다음과 같이 입력합니다.

a. pattern의 경우 모든 경로에 ****/***를 입력합니다.
b. key에 **Cache-Control**를 입력합니다.
c. value에 **s-maxage=86400**를 입력합니다.

6. 저장을 선택합니다.
7. 앱을 재배포하여 새 사용자 지정 헤더를 적용합니다.

Amplify 배포에 대한 스큐 보호

Amplify 애플리케이션에서 배포 스큐 보호를 사용하여 웹 애플리케이션의 클라이언트와 서버 간 버전 스큐 문제를 제거할 수 있습니다. Amplify 애플리케이션에 스큐 보호를 적용하면 배포 발생 시기에 관계없이 클라이언트가 항상 올바른 버전의 서버 측 자산과 상호 작용하도록 할 수 있습니다.

버전 스큐는 웹 개발자에게 일반적인 과제입니다. 웹 브라우저가 애플리케이션의 오래된 버전을 실행 중이고 서버가 새 버전을 실행 중일 때 발생합니다. 이러한 불일치로 인해 애플리케이션 사용자에게 예측할 수 없는 동작, 오류 및 성능이 저하될 수 있습니다. Amplify 배포 스큐 보호 기능은 웹 브라우저에서 실행되는 클라이언트를 특정 배포에 고정합니다. 이렇게 하면 Amplify가 항상 해당 특정 배포에 대한 자산을 제공하여 클라이언트와 서버를 동기화된 상태로 유지할 수 있습니다.

Amplify의 스큐 보호 기능은 새 배포를 릴리스할 때 애플리케이션 사용자의 오류를 줄일 수 있습니다. 또한 이전 및 이전 버전과의 호환성 문제를 관리하는 데 소요되는 시간을 줄여 개발자 경험을 개선할 수 있습니다.

스큐 보호 기능 세부 정보:

지원되는 애플리케이션 유형

Amplify가 지원하는 모든 프레임워크로 생성된 정적 및 SSR 애플리케이션에 스큐 보호를 추가할 수 있습니다. 애플리케이션은 Git 리포지토리 또는 수동 배포에서 배포할 수 있습니다.

`WEB_DYNAMIC` 플랫폼에 배포된 애플리케이션(Next.js 버전 11 이하)에는 스큐 보호를 추가할 수 없습니다.

지속 시간

정적 애플리케이션의 경우 Amplify는 1주일의 배포를 제공합니다. SSR 애플리케이션의 경우 최대 8개의 이전 배포에 대한 스큐 보호를 보장합니다.

비용

애플리케이션에 스큐 보호를 추가하는 데 드는 추가 비용은 없습니다.

성능 고려 사항

애플리케이션에 대해 스큐 보호가 활성화된 경우 Amplify는 CDN 캐시 구성을 업데이트해야 합니다. 따라서 스큐 보호를 활성화한 후 첫 번째 배포에는 최대 10분이 걸릴 것으로 예상해야 합니다.

주제

- Amplify 애플리케이션에 대한 배포 스큐 보호 구성

- 스큐 보호 작동 방식

Amplify 애플리케이션에 대한 배포 스큐 보호 구성

Amplify 콘솔 AWS Command Line Interface, 또는 SDKs. 이 기능은 브랜치 수준에서 적용됩니다. 브랜치에 대해 스큐 보호가 활성화된 후 수행되는 새 배포만 스큐 보호됩니다.

AWS CLI 또는 SDKs를 사용하여 배포 스큐 보호를 추가하거나 제거하려면 `CreateBranch.enableSkewProtection` 및 `UpdateBranch.enableSkewProtection` 필드를 사용합니다. 자세한 내용은 Amplify API 참조 설명서의 CreateBranch 및 UpdateBranch를 참조하세요.

더 이상 제공되지 않도록 특정 배포를 제거하려면 `DeleteJob` API를 사용합니다. 자세한 내용은 Amplify API 참조 설명서의 DeleteJob을 참조하세요.

현재 Amplify Hosting에 이미 배포된 애플리케이션에서만 스큐 보호를 활성화할 수 있습니다. 다음 지침에 따라 Amplify 콘솔을 사용하여 브랜치에 스큐 보호를 추가합니다.

Amplify 애플리케이션의 브랜치에 대한 스큐 보호 활성화

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/amplify/ Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 스큐 보호를 활성화할 배포된 앱의 이름을 선택합니다.
3. 탐색 창에서 앱 설정을 선택한 다음 브랜치 설정을 선택합니다.
4. 브랜치 섹션에서 업데이트할 브랜치의 이름을 선택합니다.
5. 작업 메뉴에서 스큐 보호 활성화를 선택합니다.
6. 확인 창에서 확인을 선택합니다. 이제 브랜치에 대해 스큐 보호가 활성화됩니다.
7. 애플리케이션 브랜치를 재배포합니다. 스큐 보호가 활성화된 후 이루어진 배포만 스큐 보호됩니다.

다음 지침에 따라 Amplify 콘솔을 사용하여 애플리케이션의 브랜치에서 스큐 보호를 제거합니다.

Amplify 애플리케이션의 브랜치에서 스큐 보호 제거

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/amplify/ Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 스큐 보호를 제거할 배포된 앱의 이름을 선택합니다.

3. 탐색 창에서 앱 설정을 선택한 다음 브랜치 설정을 선택합니다.
4. 브랜치 섹션에서 업데이트할 브랜치의 이름을 선택합니다.
5. 작업 메뉴에서 스큐 보호 비활성화를 선택합니다. 이제 브랜치에 대해 스큐 보호가 비활성화되고 최신 콘텐츠만 제공됩니다.

스큐 보호 작동 방식

대부분의 경우 _dpl 쿠키의 기본 동작은 스큐 보호 요구 사항을 충족합니다. 그러나 다음 고급 시나리오에서는 `X-Amplify-Dpl` 헤더 및 `dpl` 쿼리 파라미터를 사용하여 스큐 보호를 더 잘 활성화합니다.

- 동시에 여러 브라우저 탭에 웹 사이트 로드
- 서비스 작업자 사용

Amplify는 클라이언트에 제공할 콘텐츠를 결정할 때 수신되는 요청을 다음 순서로 평가합니다.

1. **`X-Amplify-Dpl`** 헤더 - 애플리케이션은이 헤더를 사용하여 요청을 특정 Amplify 배포로 보낼 수 있습니다. 이 요청 헤더는 값을 사용하여 설정할 수 있습니다`process.env.AWS_AMPLIFY_DEPLOYMENT_ID`.
2. **`dpl`** 쿼리 파라미터 - Next.js 애플리케이션은 지문으로 인쇄된 자산(.js 및 .css 파일)에 대한 요청에 대해 _dpl 쿼리 파라미터를 자동으로 설정합니다.
3. _dpl 쿠키 - 모든 스큐 보호 애플리케이션의 기본값입니다. 특정 브라우저의 경우 도메인과 상호 작용하는 모든 브라우저 탭 또는 인스턴스에 대해 동일한 쿠키가 전송됩니다.

   브라우저 탭마다 로드된 웹 사이트의 버전이 다른 경우 _dpl 쿠키는 모든 탭에서 공유됩니다. 이 시나리오에서는 _dpl 쿠키로 전체 스큐 보호를 달성할 수 없으며 스큐 보호를 위해 `X-Amplify-Dpl` 헤더 사용을 고려해야 합니다.

X-Amplify-Dpl 헤더 예제

다음 예제에서는 `X-Amplify-Dpl` 헤더를 통해 스큐 보호에 액세스하는 Next.js SSR 페이지의 코드를 보여줍니다. 페이지는 API 경로 중 하나를 기반으로 콘텐츠를 렌더링합니다. API 라우팅에 사용할 배포는의 값으로 설정된 `X-Amplify-Dpl` 헤더를 사용하여 지정됩니다`process.env.AWS_AMPLIFY_DEPLOYMENT_ID`.

```
import { useEffect, useState } from 'react';
```

```
export default function MyPage({deploymentId}) {
    const [data, setData] = useState(null);

    useEffect(() => {
        fetch('/api/hello', {
            headers: {
                'X-Amplify-Dpl': process.env.AWS_AMPLIFY_DEPLOYMENT_ID
            },
        })
        .then(res => res.json())
        .then(data => setData(data))
        .catch(error => console.error("error", error))
    }, []);

    return <div>
        {data ? JSON.stringify(data) : "Loading ... " }
    </div>
}
```

Amplify 애플리케이션 모니터링

AWS Amplify 는 호스팅 애플리케이션을 모니터링하기 위한 다음 기능을 제공합니다.

- CloudWatch 지표 - Amplify는 Amazon CloudWatch를 통해 애플리케이션의 트래픽, 오류, 데이터 전송 및 지연 시간을 모니터링하는 데 사용할 수 있는 지표를 내보냅니다.
- 액세스 로그 - Amplify는 애플리케이션에 대한 요청에 대한 자세한 정보가 포함된 액세스 로그를 제공합니다.
- CloudTrail 로깅 - Amplify는 Amplify에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 AWS CloudTrail 제공하는와 통합됩니다. CloudTrail 콘솔에서 이러한 이벤트를 볼 수 있습니다.

주제

Amazon CloudWatch를 사용하여 Amplify 애플리케이션 모니터링

AWS Amplify 는 Amazon CloudWatch와 통합되어 Amplify 애플리케이션의 지표를 거의 실시간으로 모니터링하고 지표가 설정한 임계값을 초과할 때 알림을 보내는 경보를 생성할 수 있습니다. CloudWatch 작동 방법의 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

지원되는 CloudWatch 지표

Amplify는 `AWS/AmplifyHosting` 네임스페이스에서 6개의 CloudWatch 지표를 지원하여 앱의 트래픽, 오류, 데이터 전송 및 지연 시간을 모니터링합니다. 이러한 지표는 1분 간격으로 집계됩니다. CloudWatch 모니터링 지표는 무료이며 CloudWatch 서비스 할당량에 포함되지 않습니다.

사용 가능한 모든 통계가 모든 지표에 적용되는 것은 아닙니다. 다음 표에는 가장 관련성이 높은 통계가 지원되는 각 지표에 대한 설명과 함께 나열되어 있습니다.

Metrics	설명
요청	앱에서 수신한 최종 사용자 요청의 총 수.

Metrics	설명
	가장 관련성이 높은 통계는 Sum입니다. Sum 통계를 사용하여 총 요청 수를 확인할 수 있습니다.
BytesDownloaded	GET,HEAD, 요청에 대해 시청자가 앱을 통해 전송(다운로드)한 총 데이터 양(바이트)입니다. OPTIONS 가장 관련성이 높은 통계는 Sum입니다.
BytesUploaded	헤더를 포함하여 모든 요청에 대해 앱으로 전송된(업로드된) 총 데이터 양(바이트). Amplify는 애플리케이션에 업로드된 데이터에 대해 비용을 청구하지 않습니다. 가장 관련성이 높은 통계는 Sum입니다.
4xxErrors	HTTP 상태 코드 400-499 범위에서 오류를 반환한 요청 수. 가장 관련성이 높은 통계는 Sum입니다. Sum 통계를 사용하여 오류의 총 발생 횟수를 가져옵니다.
5xxErrors	HTTP 상태 코드 500-599 범위에서 오류를 반환한 요청 수. 가장 관련성이 높은 통계는 Sum입니다. Sum 통계를 사용하여 오류의 총 발생 횟수를 가져옵니다.

Metrics	설명
지연 시간	첫 바이트까지의 시간(초). Amplify Hosting에서 요청을 수신할 때부터 네트워크에 응답을 반환할 때까지의 총 시간입니다. 여기에는 응답이 뷰어 장치에 도달하는 데 발생한 네트워크 지연 시간은 포함되지 않습니다. 가장 관련성이 높은 통계는`Average`, `Maximum`, `Minimum`, `p10`, `p50`, `p90`, `p95`, `p100`.입니다. `Average` 통계를 사용하여 예상 지연 시간을 평가합니다.

Amplify는 다음과 같은 CloudWatch 차원을 따르십시오.

차원	설명
앱	지표 데이터는 앱에서 제공합니다.
AWS 계정	지표 데이터는의 모든 앱에 제공됩니다 AWS 계정.

CloudWatch 지표 액세스

다음 절차를 사용하여 Amplify 콘솔에서 직접 CloudWatch 지표에 액세스할 수 있습니다.

> **Note**
>
> https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 지표 AWS Management Console 에 액세스할 수도 있습니다.

Amplify 콘솔에서 지표에 액세스하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 앱 지표를 확인할 서비스를 선택합니다.

3. 탐색 창에서 모니터링을 선택한 다음 지표를 선택합니다.

CloudWatch 경보 생성

Amplify 콘솔에서 특정 기준이 충족될 때 알림을 보내는 CloudWatch 경보를 생성할 수 있습니다. 경보는 단일 CloudWatch 지표를 감시하고 지표가 평가 기간에 지정된 수에 대한 임곗값을 위반할 경우, Amazon Simple Notification Service 알림을 보냅니다.

CloudWatch 콘솔에서 또는 CloudWatch API를 사용하여 지표 수학 식을 사용하는 고급 경보를 생성할 수 있습니다. 예를 들어, 4xxErrors의 비율이 세 기간 동안 연속으로 15%를 초과한 경우, 알림을 보내는 경보를 생성할 수 있습니다. 자세한 정보는 Amazon CloudWatch 사용 설명서에서 지표 수학 표현식을 기반으로 CloudWatch 경보 생성을 참조하십시오.

경보에는 표준 CloudWatch 요금이 적용됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

Amplify 콘솔에서 다음 절차에 따라 경고를 만듭니다.

Amplify 지표에 대한 CloudWatch 경보를 생성하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 경보를 설정할 앱을 선택합니다.
3. 탐색 창에서 모니터링을 선택한 다음 경보를 선택합니다.
4. 경보 페이지에서 경보 생성을 선택합니다.
5. 경보 생성 창에서 다음과 같이 경보를 구성합니다.

    a. 지표의 경우, 목록에서 모니터링할 지표의 이름을 선택합니다.

    b. 경보 이름에 경보 이름을 입력합니다. 예를 들어 요청을 모니터링하는 경우, 경보 이름을 지정할 수 **`HighTraffic`** 있습니다. 이름은 ASCII 문자만 포함해야 합니다.

    c. 알림 설정에 다음 중 하나를 수행하십시오.

    - i. 새 Amazon SNS 주제를 설정하려면 새로 만들기 를 선택합니다.
      ii. 이메일 주소에는 알림 수신자의 이메일 주소를 입력합니다.
      iii. 새 이메일 주소 추가를 선택하여 수신자를 더 추가합니다.
    - i. Amazon SNS 주제를 재사용하려면 기존 항목 을 선택합니다.
      ii. SNS 항목에 대해서는 목록에서 기존 Amazon SNS 항목의 이름을 선택합니다.

    d. 메트릭의 통계 예시에서 알람 조건을 다음과 같이 설정하십시오.

i. 지표가 임곗값보다 크거나, 작거나, 같아야 하는지 여부를 지정합니다.

ii. 임곗값을 지정합니다.

iii. 경보가 간접적으로 호출되려면 경보 상태에 있어야 하는 연속 평가 기간 수를 지정합니다.

iv. 평가 간격 시간의 길이를 지정합니다.

e. 확인을 선택합니다.

> **Note**
>
> 지정한 각 Amazon SNS 수신자는 AWS 알림으로부터 확인 이메일을 수신합니다. 이메일에는 수신자가 구독을 확인하고 알림을 받기 위해 따라야 하는 링크가 포함되어 있습니다.

SSR 앱용 CloudWatch Logs에 액세스

Amplify는 SSR 런타임에 대한 정보를 사용자 AWS 계정의 Amazon CloudWatch Logs로 보냅니다. Amplify Hosting 컴퓨팅에 SSR 앱을 배포하는 경우 앱을 사용하려면 Amplify가 사용자를 대신하여 다른 서비스를 호출할 때 수임하는 IAM 서비스 역할이 필요합니다. Amplify Hosting 컴퓨팅이 자동으로 서비스 역할을 생성하도록 허용하거나 사용자가 생성한 역할을 지정할 수 있습니다.

Amplify에서 IAM 역할을 생성하도록 허용하면 해당 역할에 CloudWatch Logs를 생성할 권한이 부여됩니다. IAM 역할을 직접 생성하는 경우, Amplify가 Amazon CloudWatch Logs에 액세스할 수 있도록 하려면 정책에 다음 권한을 추가해야 합니다.

```
logs:CreateLogStream
logs:CreateLogGroup
logs:DescribeLogGroups
logs:PutLogEvents
```

서비스 역할 추가에 대한 자세한 내용은 섹션을 참조하세요백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가. 서버측 렌더링된 앱 배포에 대한 자세한 내용은 Amplify Hosting을 통해 서버 측 렌더링 애플리케이션 배포을 참조하십시오.

CloudWatch 콘솔 또는 Amplify 콘솔에서 SSR 애플리케이션에 대한 Amplify Hosting 컴퓨팅 로그를 볼 수 있습니다. Amplify 콘솔에서 로그를 보려면 다음 지침을 따르십시오.

Amplify 콘솔에서 SSR 애플리케이션에 대한 CloudWatch 로그를 보려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. CloudWatch 로그를 볼 SSR 앱을 선택합니다.
3. 탐색 창에서 모니터링을 선택한 다음 컴퓨팅 로그 호스팅을 선택합니다.
4. 컴퓨팅 로그 호스팅 페이지에서 특정 브랜치에 대한 CloudWatch 로그 그룹을 검색하고 선택합니다.

Amplify 애플리케이션의 액세스 로그 검색 및 분석

Amplify는 Amplify에서 호스팅하는 모든 앱에 대한 액세스 로그를 저장합니다. 액세스 로그는 호스팅된 앱에 대해 이루어진 요청에 대한 정보를 포함합니다. Amplify는 사용자가 앱을 삭제할 때까지 앱에 대한 모든 액세스 로그를 유지합니다. 앱에 대한 모든 액세스 로그는 Amplify 콘솔에서 사용할 수 있습니다. 그러나 액세스 로그에 대한 개별 요청은 사용자가 지정하는 2주 기간으로 제한됩니다.

> **Warning**
>
> URLs에 비밀, 자격 증명 또는 민감한 데이터를 경로 또는 쿼리 파라미터로 포함하지 마십시오. 이러한 값은 Amplify 애플리케이션의 액세스 로그에서 일반 텍스트로 볼 수 있습니다.

Amplify는 고객 간 CloudFront 배포를 재사용하지 않습니다. Amplify는 CloudFront 배포를 미리 생성하므로 새 앱을 배포할 때 CloudFront 배포가 생성될 때까지 기다릴 필요가 없습니다. 이러한 배포는 Amplify 앱에 할당되기 전에 봇으로부터 트래픽을 수신할 수 있습니다. 하지만 할당되기 전에는 항상 찾을 수 없음으로 응답하도록 구성되어 있습니다. 앱 액세스 로그에 앱을 만들기 전 일정 기간의 항목이 포함되어 있는 경우, 이러한 항목은 이 활동과 관련이 있습니다.

> **Important**
>
> 모든 요청을 완전히 살펴보기 보다는 콘텐츠에 대한 요청 특성을 이해하는 데 로그를 사용하는 것이 좋습니다. Amplify는 최대 효과에 기초하여 액세스 로그를 전송합니다. 요청에 따라서는 실제로 요청이 처리된 지 한참 후에 로그 레코드가 전송되거나 아예 전송되지 않을 수도 있습니다. 액세스 로그에서 로그 항목을 생략하면 액세스 로그의 항목 수가 AWS 결제 및 사용 보고서에 표시되는 사용량과 일치하지 않습니다.

앱의 액세스 로그 검색

Amplify 앱에 대한 액세스 로그를 검색하려면 다음 절차를 사용합니다.

액세스 로그를 보기

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 액세스 로그를 보려는 앱을 선택합니다.
3. 탐색 창에서 모니터링을 선택한 다음 액세스 로그를 선택합니다.
4. 시간 범위 편집을 선택합니다.
5. 시간 범위 편집 창에서 다음을 수행합니다.
   a. 시작 날짜에 로그를 검색할 2주 간격의 첫 번째 요일을 지정합니다.
   b. 시작 시간에서 로그 검색을 시작할 첫날의 시간을 선택합니다.
   c. 확인을 선택합니다.
6. Amplify 콘솔은 액세스 로그 섹션에 지정된 시간 범위의 로그를 표시합니다. 다운로드를 선택하여 로그를 CSV 형식으로 저장합니다.

액세스 로그 분석

Amazon S3 버킷에 CSV 파일을 저장하여 액세스 로그를 분석할 수 있습니다. 액세스 로그를 분석하는 한 가지 방법은 Athena를 사용하는 것입니다. Athena는 서비스에 대한 데이터를 분석하는 데 도움이 되는 대화형 쿼리 AWS 서비스입니다. 여기의 단계별 지침에 따라 테이블을 생성할 수 있습니다. 테이블이 생성되면 다음과 같이 데이터를 쿼리할 수 있습니다.

```
SELECT SUM(bytes) AS total_bytes
FROM logs
WHERE "date" BETWEEN DATE '2018-06-09' AND DATE '2018-06-11'
LIMIT 100;
```

AWS CloudTrail를 사용하여 Amplify API 호출 로깅

AWS Amplify 는 Amplify에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 Amplify에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Amplify 콘솔로부터의 호출과 Amplify API 작업에 대한 코드 호출이 포함됩

니다. 추적을 생성하면 Amplify 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail이 수집하는 정보를 사용하여 Amplify에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Amplify 정보

CloudTrail은 기본적으로 AWS 계정에서 활성화됩니다. Amplify에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은AWS CloudTrail 사용 설명서에서 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

Amplify에 대한 이벤트를 포함하여 AWS 계정의 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 지역에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 작업하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 AWS CloudTrail User Guide의 다음 섹션을 참조하십시오.

- AWS 계정에 대한 추적 생성
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 Amplify 작업은 CloudTrail에 기록되며 AWS Amplify 콘솔 API 참조, AWS Amplify 관리자 UI API 참조 및 Amplify UI 빌더 API 참조에 문서화되어 있습니다. 예를 들어, `CreateApp`, `DeleteApp` 및 `DeleteBackendEnvironment` 작업에 대한 직접 호출은 CloudTrail 로그 파일의 항목을 생성합니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 요청이 이루어졌습니까?
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었습니다.
- 다른 AWS 서비스에서 요청한 경우.

자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail userIdentity 요소를 참조하십시오.

Athena 로그 파일 항목의 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예제에서는 AWS Amplify 콘솔 API 참조 `ListApps` 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::444455556666:user/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-01-12T05:48:10Z"
            }
        }
    },
    "eventTime": "2021-01-12T06:47:29Z",
    "eventSource": "amplify.amazonaws.com",
    "eventName": "ListApps",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.898
Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.275-b01
java/1.8.0_275 vendor/Oracle_Corporation",
    "requestParameters": {
        "maxResults": "100"
    },
    "responseElements": null,
```

```
    "requestID": "1c026d0b-3397-405a-95aa-aa43aexample",
    "eventID": "c5fca3fb-d148-4fa1-ba22-5fa63example",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "444455556666"
}
```

다음 예제에서는 AWS Amplify 관리자 UI API 참조 ListBackendJobs 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::444455556666:user/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-01-13T00:47:25Z"
            }
        }
    },
    "eventTime": "2021-01-13T01:15:43Z",
    "eventSource": "amplifybackend.amazonaws.com",
    "eventName": "ListBackendJobs",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.898
 Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.275-b01
 java/1.8.0_275 vendor/Oracle_Corporation",
    "requestParameters": {
        "appId": "d23mv2oexample",
        "backendEnvironmentName": "staging"
    },
    "responseElements": {
```

```
        "jobs": [
            {
                "appId": "d23mv2oexample",
                "backendEnvironmentName": "staging",
                "jobId": "ed63e9b2-dd1b-4bf2-895b-3d5dcexample",
                "operation": "CreateBackendAuth",
                "status": "COMPLETED",
                "createTime": "1610499932490",
                "updateTime": "1610500140053"
            },
            {
                "appId": "d23mv2oexample",
                "backendEnvironmentName": "staging",
                "jobId": "06904b10-a795-49c1-92b7-185dfexample",
                "operation": "CreateBackend",
                "status": "COMPLETED",
                "createTime": "1610499657938",
                "updateTime": "1610499704458"
            }
        ],
        "appId": "d23mv2oexample",
        "backendEnvironmentName": "staging"
    },
    "requestID": "7adfabd6-98d5-4b11-bd39-c7deaexample",
    "eventID": "68769310-c96c-4789-a6bb-68b52example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "444455556666"
}
```

Amplify 애플리케이션에서 IAM 역할 사용

IAM 역할은 특정 권한이 있는 IAM 자격 증명입니다. 역할의 권한에 따라 자격 증명이 수행할 수 있는 작업과 수행할 수 없는 작업이 결정됩니다 AWS. 에서 IAM 역할을 생성하고 이를 AWS 계정 사용하여 Amplify Hosting에 권한을 위임할 수 있습니다. 역할에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할을 참조하세요.

다음 유형의 IAM 역할을 사용하여 Amplify Hosting에 사용자를 대신하여 작업을 수행하거나 다른 AWS 리소스에 액세스하는 컴퓨팅 코드를 실행하는 데 필요한 권한을 부여할 수 있습니다.

IAM 서비스 역할

Amplify는 사용자를 대신하여 작업을 수행하기 위해이 역할을 수임합니다. 이 역할은 백엔드 리소스가 있는 애플리케이션에 필요합니다.

IAM SSR 컴퓨팅 역할

서버 측 렌더링(SSR) 애플리케이션이 특정 AWS 리소스에 안전하게 액세스할 수 있도록 허용합니다.

IAM SSR CloudWatch Logs 역할

SSR 앱을 배포할 때 앱에는 Amplify가 Amazon CloudWatch Logs에 액세스할 수 있도록 Amplify가 수임하는 IAM 서비스 역할이 필요합니다.

주제

- 백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가
- AWS 리소스에 대한 액세스를 허용하는 SSR 컴퓨팅 역할 추가
- CloudWatch Logs에 액세스할 수 있는 권한이 있는 서비스 역할 추가

백엔드 리소스를 배포할 수 있는 권한이 있는 서비스 역할 추가

Amplify에는 프런트엔드와 함께 백엔드 리소스를 배포할 수 있는 권한이 필요합니다. 서비스 역할을 사용하여 이를 수행합니다. 서비스 역할은 Amplify Hosting에 사용자를 대신하여 백엔드를 배포, 생성 및 관리할 수 있는 권한을 제공하는 AWS Identity and Access Management (IAM) 역할입니다.

IAM 서비스 역할이 필요한 새 앱을 생성할 때 Amplify Hosting이 자동으로 서비스 역할을 생성하도록 허용하거나 이미 생성한 IAM 역할을 선택할 수 있습니다. 이 섹션에서는 계정 관리 권한이 있고

Amplify 애플리케이션이 백엔드를 배포, 생성 및 관리하는 데 필요한 리소스에 직접 액세스할 수 있도록 명시적으로 허용하는 Amplify 서비스 역할을 생성하는 방법을 알아봅니다.

IAM 콘솔에서 Amplify 서비스 역할 생성

서비스 역할을 생성하는 방법

1. IAM 콘솔을 열고 왼쪽 탐색 모음에서 역할을 선택한 다음, 역할 생성을 선택합니다.
2. 신뢰할 수 있는 엔터티 선택 페이지에서 AWS 서비스를 선택합니다. 사용 사례에서 Amplify - 백엔드 배포를 선택한 후 다음을 선택합니다.
3. 권한 추가 페이지에서 다음을 선택합니다.
4. 이름, 보기 및 생성 페이지에서 역할 이름에 **AmplifyConsoleServiceRole-AmplifyRole**과 같은 의미 있는 이름을 입력합니다.
5. 모든 기본값을 수락하고 역할 생성을 선택합니다.
6. Amplify 콘솔로 돌아가 역할을 앱에 연결합니다.
   - 새 앱을 배포하는 경우 다음을 수행합니다.
     a. 서비스 역할 목록을 새로 고칩니다.
     b. 방금 생성한 역할을 선택합니다. 이 예제에서는 AmplifyConsoleServiceRole-AmplifyRole과 비슷해야 합니다.
     c. 다음을 선택하고 단계에 따라 앱 배포를 완료합니다.
   - 기존 앱이 있는 경우 다음을 수행합니다.
     a. 탐색 창에서 앱 설정을 선택한 다음 IAM 역할을 선택합니다.
     b. IAM 역할 페이지의 서비스 역할 섹션에서 편집을 선택합니다.
     c. 서비스 역할 페이지의 서비스 역할 목록에서 방금 생성한 역할을 선택합니다.
     d. 저장(Save)을 선택합니다.
7. Amplify에는 이제 앱에 백엔드 리소스를 배포할 수 있는 권한이 있습니다.

혼동된 대리자를 방지하기 위해 서비스 역할의 신뢰 정책 편집

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 자세한 내용은 교차 서비스 혼동된 대리자 방지 단원을 참조하십시오.

현재 Amplify-Backend Deployment 서비스 역할에 대한 기본 신뢰 정책은 aws:SourceArn 및 aws:SourceAccount 글로벌 조건 컨텍스트 키를 적용하여 혼동된 대리자를 방지합니다. 하지만 이전에 Amplify-Backend Deployment 역할을 계정에 생성한 경우에는 역할의 신뢰 정책을 업데이트하여 이러한 조건을 추가함으로써 혼동된 대리자를 방지할 수 있습니다.

다음 예제를 사용하여 계정의 앱에 대한 액세스를 제한할 수 있습니다. 예제에서 리전 및 애플리케이션 ID를 실제 정보로 바꿉니다.

```
"Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
```

를 사용하여 역할의 신뢰 정책을 편집하는 방법에 대한 지침은 IAM 사용 설명서의 역할 수정(콘솔)을 AWS Management Console참조하세요.

AWS 리소스에 대한 액세스를 허용하는 SSR 컴퓨팅 역할 추가

이 통합을 통해 Amplify SSR Compute 서비스에 IAM 역할을 할당하여 서버 측 렌더링(SSR) 애플리케이션이 역할의 권한에 따라 특정 AWS 리소스에 안전하게 액세스할 수 있습니다. 예를 들어 할당된 IAM 역할에 정의된 권한에 따라 앱의 SSR 컴퓨팅 함수가 Amazon Bedrock 또는 Amazon S3 버킷과 같은 다른 AWS 서비스 또는 리소스에 안전하게 액세스하도록 허용할 수 있습니다.

IAM SSR 컴퓨팅 역할은 임시 자격 증명을 제공하므로 환경 변수에서 수명이 긴 보안 자격 증명을 하드코딩할 필요가 없습니다. IAM SSR 컴퓨팅 역할을 사용하면 가능한 경우 최소 권한 부여 및 단기 자격 증명 사용의 AWS 보안 모범 사례에 부합합니다.

이 섹션 뒷부분의 지침에서는 사용자 지정 권한이 있는 정책을 생성하고 정책을 역할에 연결하는 방법을 설명합니다. 역할을 생성할 때 Amplify에 역할을 수임할 수 있는 권한을 부여하는 사용자 지정 신뢰 정책을 연결해야 합니다. 신뢰 관계가 올바르게 정의되지 않은 경우 역할을 추가하려고 할 때 오류가 발생합니다. 다음 사용자 지정 신뢰 정책은 Amplify에 역할을 수임할 수 있는 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "amplify.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Amplify 콘솔, AWS SDKs 또는를 사용하여의 IAM 역할을 기존 SSR 애플리케이션 AWS 계정 과 연결할 수 있습니다 AWS CLI. 연결하는 역할은 Amplify SSR 컴퓨팅 서비스와 자동으로 연결되어 다른 AWS 리소스에 액세스할 수 있도록 지정한 권한을 부여합니다. 시간이 지남에 따라 애플리케이션의 요구 사항이 변경되면 애플리케이션을 재배포하지 않고도 연결된 IAM 역할을 수정할 수 있습니다. 이를 통해 유연성을 제공하고 애플리케이션 가동 중지 시간을 줄일 수 있습니다.

> **Important**
>
> 보안 및 규정 준수 목표에 맞게 애플리케이션을 구성해야 합니다. 여기에는 사용 사례를 지원하는 데 필요한 최소 권한 세트를 갖도록 구성해야 하는 SSR 컴퓨팅 역할 관리가 포함됩니다. 자세한 내용은 IAM SSR 컴퓨팅 역할 보안 관리 단원을 참조하십시오.

IAM 콘솔에서 SSR 컴퓨팅 역할 생성

IAM SSR Compute 역할을 Amplify 애플리케이션에 연결하려면 먼저 역할이에 이미 있어야 합니다 AWS 계정. 이 섹션에서는 IAM 정책을 생성하고 Amplify가 특정 AWS 리소스에 액세스하기 위해 수임할 수 있는 역할에 연결하는 방법을 알아봅니다.

IAM 역할을 생성할 때 최소 권한 부여 AWS 모범 사례를 따르는 것이 좋습니다. IAM SSR 컴퓨팅 역할은 SSR 컴퓨팅 함수에서만 호출되므로 코드를 실행하는 데 필요한 권한만 부여해야 합니다.

AWS Management Console AWS CLI또는 SDKs를 사용하여 IAM에서 정책을 생성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 고객 관리형 정책을 사용하여 사용자 지정 IAM 권한 정의를 참조하세요.

다음 지침은 IAM 콘솔을 사용하여 Amplify Compute 서비스에 부여할 권한을 정의하는 IAM 정책을 생성하는 방법을 보여줍니다.

IAM 콘솔 JSON 정책 편집기를 사용하여 정책을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택합니다.
4. 정책 편집기 섹션에서 JSON 옵션을 선택합니다.
5. JSON 정책 문서를 입력하거나 붙여 넣습니다.
6. 정책에 권한 추가를 완료했으면 다음을 선택합니다.
7. 검토 및 생성 페이지에서 생성하는 정책에 대한 정책 이름과 설명(선택 사항)을 입력합니다. 이 정책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인합니다.
8. 정책 생성을 선택하고 새로운 정책을 저장합니다.

정책을 생성한 후 다음 지침에 따라 정책을 IAM 역할에 연결합니다.

Amplify에 특정 AWS 리소스에 대한 권한을 부여하는 역할을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. 사용자 지정 신뢰 정책(Custom trust policy) 역할 유형을 선택합니다.
4. 사용자 지정 신뢰 정책 섹션에서 역할에 대한 사용자 지정 신뢰 정책을 입력합니다. 역할 신뢰 정책이 필요하며 역할을 수임하기 위해 신뢰할 수 있는 보안 주체를 정의합니다.

   다음 신뢰 정책을 복사하여 붙여 넣어 Amplify 서비스에이 역할을 수임할 수 있는 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "amplify.amazonaws.com"
                ]
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

5. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.
6. 권한 추가 페이지에서 이전 절차에서 생성한 정책의 이름을 검색하고 선택합니다. 그런 다음 다음을 선택합니다.
7. 역할 이름(Role name)에 역할 이름을 입력합니다. 역할 이름은 내에서 고유해야 합니다 AWS 계정. 대소문자는 구별하지 않습니다. 예를 들어, 이름이 **PRODROLE**과 **prodrole**, 두 가지로 지정된 역할을 만들 수는 없습니다. 다른 AWS 리소스가 역할을 참조할 수 있으므로 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.
8. (선택 사항)설명에 새 역할에 대한 설명을 입력합니다.
9. (선택 사항) 1단계: 신뢰할 수 있는 엔터티 선택 또는 2단계: 권한 추가 섹션에서 편집을 선택하여 역할의 사용자 지정 정책과 권한을 편집합니다.
10. 역할을 검토한 다음 역할 생성을 선택합니다.

Amplify 앱에 IAM SSR 컴퓨팅 역할 추가

에서 IAM 역할을 생성한 후 Amplify 콘솔의 앱과 연결할 AWS 계정수 있습니다.

Amplify 콘솔의 앱에 SSR 컴퓨팅 역할을 추가하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/amplify/ Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 컴퓨팅 역할을 추가할 앱의 이름을 선택합니다.
3. 탐색 창에서 앱 설정을 선택한 다음 IAM 역할을 선택합니다.
4. 컴퓨팅 역할 섹션에서 편집을 선택합니다.
5. 기본 역할 목록에서 연결하려는 역할의 이름을 검색하고 선택합니다. 이 예제에서는 이전 절차에서 생성한 역할의 이름을 선택할 수 있습니다. 기본적으로 선택한 역할은 앱의 모든 브랜치와 연결됩니다.

   역할의 신뢰 관계가 올바르게 정의되지 않으면 오류가 발생하고 역할을 추가할 수 없습니다.
6. (선택 사항) 애플리케이션이 퍼블릭 리포지토리에 있고 자동 브랜치 생성을 사용하거나 풀 요청에 대한 웹 미리 보기가 활성화된 경우 앱 수준 역할을 사용하지 않는 것이 좋습니다. 대신 컴퓨팅 역

할은 특정 리소스에 액세스해야 하는 브랜치에만 연결합니다. 기본 앱 수준 동작을 재정의하고 역할을 특정 브랜치에 연결하려면 다음을 수행합니다.

a. 브랜치에서 사용할 브랜치의 이름을 선택합니다.

b. 컴퓨팅 역할에서 브랜치와 연결할 역할의 이름을 선택합니다.

7. 저장을 선택합니다.

IAM SSR 컴퓨팅 역할 보안 관리

보안은 AWS 와 사용자 간의 공동 책임입니다. 보안 및 규정 준수 목표에 맞게 애플리케이션을 구성해야 합니다. 여기에는 사용 사례를 지원하는 데 필요한 최소 권한 세트를 갖도록 구성해야 하는 SSR 컴퓨팅 역할 관리가 포함됩니다. 지정한 SSR 컴퓨팅 역할에 대한 자격 증명은 SSR 함수의 런타임에 즉시 사용할 수 있습니다. SSR 코드가 버그로 인해 의도적으로 또는 원격 코드 실행(RCE)을 허용하여 이러한 자격 증명을 노출하는 경우 권한이 없는 사용자는 SSR 역할 및 권한에 액세스할 수 있습니다.

퍼블릭 리포지토리의 애플리케이션이 풀 요청에 SSR 컴퓨팅 역할과 자동 브랜치 생성 또는 웹 미리 보기를 사용하는 경우 역할에 액세스할 수 있는 브랜치를 신중하게 관리해야 합니다. 앱 수준 역할은 사용하지 않는 것이 좋습니다. 대신 분기 수준에서 컴퓨팅 역할을 연결해야 합니다. 이를 통해 특정 리소스에 대한 액세스가 필요한 브랜치에만 권한을 부여할 수 있습니다.

역할의 자격 증명이 노출되면 다음 작업을 수행하여 역할의 자격 증명에 대한 모든 액세스를 제거합니다.

1. 모든 세션 취소

   역할의 자격 증명에 대한 모든 권한을 즉시 취소하는 방법에 대한 지침은 IAM 역할 임시 보안 자격 증명 취소를 참조하세요.

2. Amplify 콘솔에서 역할 삭제

   이 작업은 즉시 적용됩니다. 애플리케이션을 재배포할 필요가 없습니다.

Amplify 콘솔에서 컴퓨팅 역할을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/amplify/ Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 컴퓨팅 역할을 제거할 앱의 이름을 선택합니다.
3. 탐색 창에서 앱 설정을 선택한 다음 IAM 역할을 선택합니다.

4. 컴퓨팅 역할 섹션에서 편집을 선택합니다.
5. 기본 역할을 삭제하려면 역할 이름 오른쪽에 있는 X를 선택합니다.
6. 저장(Save)을 선택합니다.

CloudWatch Logs에 액세스할 수 있는 권한이 있는 서비스 역할 추가

Amplify는 SSR 런타임에 대한 정보를 사용자 AWS 계정의 Amazon CloudWatch Logs로 보냅니다. SSR 앱을 배포할 때 Amplify가 사용자를 대신하여 다른 서비스를 호출할 때 맡는 IAM 서비스 역할이 앱에 필요합니다. Amplify Hosting 컴퓨팅이 자동으로 서비스 역할을 생성하도록 허용하거나 사용자가 생성한 역할을 지정할 수 있습니다.

Amplify에서 IAM 역할을 생성하도록 허용하면 해당 역할에 CloudWatch Logs를 생성할 권한이 부여됩니다. IAM 역할을 직접 생성하는 경우, Amplify가 Amazon CloudWatch Logs에 액세스할 수 있도록 하려면 정책에 다음 권한을 추가해야 합니다.

```
logs:CreateLogStream
logs:CreateLogGroup
logs:DescribeLogGroups
logs:PutLogEvents
```

Git 리포지토리용 통합 웹후크

Amplify Hosting은 웹후크를 사용하여 Git 리포지토리에 대한 새 커밋 후 빌드를 자동으로 시작합니다. 통합 웹후크 기능은 Amplify와 Git 공급자의 통합을 개선하고 더 많은 Amplify 애플리케이션을 단일 리포지토리에 연결할 수 있도록 합니다. 통합 웹후크를 통해 Amplify는 이제 리포지토리의 연결된 모든 애플리케이션에 대해 리전당 단일 웹후크를 사용합니다. 예를 들어 리포지토리가 미국 동부(버지니아 북부) 및 미국 서부(오레곤) 리전의 애플리케이션에 연결된 경우 두 개의 통합 웹후크가 있습니다.

이 릴리스 이전에 Amplify는 리포지토리와 연결된 각 앱에 대해 새 웹후크를 생성했습니다. 단일 리포지토리에 여러 앱이 있는 경우 개별 Git 공급자가 적용하는 웹후크 제한에 도달하여 앱을 더 추가할 수 없습니다. 이는 단일 리포지토리에 여러 프로젝트가 있는 모노리포지토리에서 작업하는 팀에 특히 어려운 일이었습니다.

통합 웹후크는 다음과 같은 이점을 제공합니다.

- Git 공급자 웹후크 제한 극복: 단일 리포지토리에 필요한 만큼 Amplify 앱을 연결할 수 있습니다.
- 향상된 모노 리포지토리 지원: 여러 프로젝트가 단일 리포지토리를 공유하는 모노 리포지토리로 작업할 때 유연성과 효율성이 향상됩니다.
- 간소화된 관리: 단일 리포지토리 웹후크로 여러 Amplify 앱을 관리하면 복잡성과 잠재적 장애 지점이 줄어듭니다.
- 워크플로 통합 개선: Git 공급자가 할당한 웹후크를 개발 프로세스의 다른 필수 워크플로에 사용할 수 있습니다.

통합 웹후크 시작하기

새 앱 생성

Git 리포지토리에서 Amplify Hosting에 새 애플리케이션을 배포하면 리포지토리에 통합 웹후크 기능이 자동으로 구현됩니다. 새 애플리케이션 생성에 대한 지침은 섹션을 참조하세요Amplify Hosting에 앱 배포 시작하기.

기존 앱 업데이트

기존 Amplify 애플리케이션의 경우 기존 웹후크를 통합 웹후크로 바꾸려면 Git 리포지토리를 애플리케이션에 다시 연결해야 합니다. Git 공급자가 허용하는 최대 웹후크 수에 이미 도달한 경우 통합 웹후크로 마이그레이션하는 데 실패할 수 있습니다. 이 경우 다시 연결하기 전에 기존 웹후크를 하나 이상 수동으로 제거합니다.

리포지토리에 여러 AWS 리전에 배포되는 여러 애플리케이션이 있을 수 있습니다. Amplify 작업은 리전을 기반으로 하므로 통합 웹후크로의 마이그레이션은 Amplify 앱을 다시 연결한 리전의 웹후크에 대해서만 수행됩니다. 따라서 리포지토리에 애플리케이션 ID 기반 웹후크와 리전 기반 통합 웹후크가 모두 표시될 수 있습니다.

다음 지침에 따라 기존 Amplify 앱을 통합 웹후크로 마이그레이션합니다.

기존 Amplify 앱을 통합 웹후크로 마이그레이션하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 통합 웹후크로 마이그레이션할 앱을 선택합니다.
3. 탐색 창에서 앱 설정을 선택한 다음 브랜치 설정을 선택합니다.
4. 브랜치 설정 페이지에서 리포지토리 재연결을 선택합니다.
5. 통합 웹후크로 성공적으로 마이그레이션되었는지 확인하려면 Git 리포지토리의 웹후크 설정으로 이동합니다. 형식의 단일 웹후크 URL이 표시되어야 합니다`https://amplify-webhooks.`*`Region`*`.amazonaws.com/`*`git-provider`*.

Amplify의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 제공 범위 내 서비스를 AWS Amplify참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amplify 사용 시 책임 분담 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 항목에서는 보안 및 규정 준수 목표를 충족하도록 Amplify를 구성하는 방법을 보여줍니다. 또한 Amplify 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

Amplify의 ID 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 누가 Amplify 리소스를 사용하도록 인증되

고(로그인) 권한이 부여되는지(권한 보유)를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 Amplify에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 – Amplify 서비스를 사용하여 작업을 수행하는 경우, 필요한 자격 증명과 권한을 관리자가 제공합니다. 다른 Amplify 기능을 사용하여 작업을 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. Amplify의 기능에 액세스할 수 없다면 Amplify 자격 증명 및 액세스 문제 해결 부분을 참조하십시오.

서비스 관리자 – 회사에서 Amplify 리소스를 책임지고 있다면 Amplify에 대한 완전한 액세스 권한이 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amplify 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 Amplify에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amplify에서 IAM을 사용하는 방법 섹션을 참조하십시오.

IAM 관리자 - IAM 관리자라면 Amplify에 대한 액세스 권한을 관리하는 정책을 작성하는 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 Amplify 자격 증명 기반 정책 예제를 보려면 Amplify의 자격 증명 기반 정책 예 섹션을 참조하십시오.

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. AWS 계정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 다중 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스 에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, AWS Directory Service, Identity Center 디렉터리 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스 에 액세스하는 모든 사용자의 사용자입니다. 페더레이션 자격 증명에 액세스할 때 역할을 AWS 계정수임하고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 및 애플리케이션에서 사용할 수 있도록 자체 ID 소스의

사용자 AWS 계정 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇인가요?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수임하려면 사용자에서 IAM 역할(콘솔)로 전환할 AWS Management Console수 있습니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.

- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 정책을 리소스에 직접 연결할 AWS 서비스수 있습니다(역할을 프록시로 사용하는 대신). 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하기 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
  - 서비스 연결 역할 - 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결된 AWS 경우 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console AWS CLI, 또는 API에서 역할 정보를 가져올 수 있습니다 AWS .

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는이 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCPs) - SCPs는의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 비즈니스가 소유한 여러 AWS 계정 를 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.
- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 AWS 계정 루트 사용자관계없이를 포함한 자격 증명에 대한 유효 권한에 영향을 미칠 수 있습니다. RCP를 AWS 서비스 지원하는 목록을 포함하여 조직 및 RCPs에 대한 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCPs)을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다.

이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amplify에서 IAM을 사용하는 방법

IAM을 사용하여 Amplify에 대한 액세스를 관리하기 전에 Amplify와 함께 사용할 수 있는 IAM 기능을 알아보십시오.

Amplify를 통해 사용할 수 있는 IAM 기능

IAM 기능	Amplify 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACLs	아니요
ABAC(정책 내 태그)	부분
임시 자격 증명	예
전달 액세스 세션(FAS)	예
서비스 역할	예
서비스 연결 역할	아니요

Amplify 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방식을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Amplify의 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. ID 기반 정책에서는 위탁자가 연결된 사용자 또는 역할에 적용되므로 위탁자를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

Amplify의 자격 증명 기반 정책 예

Amplify 자격 증명 기반 정책 예제를 보려면 Amplify의 자격 증명 기반 정책 예 섹션을 참조하십시오.

Amplify 내 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는이 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 위탁자로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 다른 경우 신뢰할 수 AWS 계정있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에도 리소스에 액세스할 수 있는 권한을 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 위탁자에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

Amplify에 대한 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action` 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

ACM 작업 목록을 보려면 서비스 권한 부여 참조의 AWS Amplify에서 정의한 작업을 참조하십시오.

Amplify의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
amplify
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "amplify:action1",
      "amplify:action2"
         ]
```

Amplify 자격 증명 기반 정책 예제를 보려면 Amplify의 자격 증명 기반 정책 예 섹션을 참조하십시오.

Amplify 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amplify 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조의 AWS Amplify에서 정의한 리소스를 참조하십시오. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS Amplify가 정의한 작업을 참조하십시오.

Amplify 자격 증명 기반 정책 예제를 보려면 Amplify의 자격 증명 기반 정책 예 섹션을 참조하십시오.

Amplify의 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 `OR` 작업을 사용하여 조건을 AWS 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amplify 조건 키 목록을 보려면 서비스 권한 부여 참조의 AWS Amplify을(를) 위한 조건 키를 참조하십시오. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 에서 정의한 작업을 AWS Amplify 참조하세요.

Amplify 자격 증명 기반 정책 예제를 보려면 Amplify의 자격 증명 기반 정책 예 섹션을 참조하십시오.

Amplify의 액세스 제어 목록(ACL)

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amplify의 ABAC(속성 기반 액세스 제어)

ABAC 지원(정책의 태그): 부분적

속성 기반 액세스 제어(ABAC)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. 에서는 AWS이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS 리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 위탁자의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Amplify에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

일부 AWS 서비스 는 임시 자격 증명을 사용하여 로그인할 때 작동하지 않습니다. 임시 자격 증명으로 AWS 서비스 작업하는를 포함한 추가 정보는 AWS 서비스 IAM 사용 설명서의 IAM으로 작업하는 섹션을 참조하세요.

사용자 이름 및 암호를 제외한 방법을 AWS Management Console 사용하여에 로그인하는 경우 임시 자격 증명을 사용합니다. 예를 들어 회사의 SSO(Single Sign-On) 링크를 AWS 사용하여에 액세스하면 해당 프로세스가 임시 자격 증명을 자동으로 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 자격 증명을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 사용자에서 IAM 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다. 그런 다음 이러한 임시 자격 증명을 사용하여 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 access AWS. AWS recommends에 액세스할 수 있습니다. 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하세요.

Amplify의 전달 액세스 세션

전달 액세스 세션(FAS) 지원: 예

IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하기 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Amplify의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amplify 기능이 중단될 수 있습니다. Amplify에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Amplify에 대한 서비스 연결 역할

서비스 링크 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM 사용 설명서의 IAM으로 작업하는AWS 서비스 섹션을 참조하십시오. 서비스 연결 역할 열에서 Yes이(가) 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amplify의 자격 증명 기반 정책 예

기본적으로 사용자 및 역할은 Amplify 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 포함하여 Amplify에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조에서 AWS Amplify에 대한 작업, 리소스 및 조건 키를 참조하십시오.

주제

- 정책 모범 사례
- Amplify 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amplify 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서

비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.

- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

Amplify 콘솔 사용

AWS Amplify 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은에서 Amplify 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

Amplify Studio가 릴리스됨에 따라 앱 또는 백엔드를 삭제하려면 `amplify` 및 `amplifybackend` 권한이 모두 필요합니다. IAM 정책이 `amplify` 권한만 제공하는 경우, 사용자가 앱을 삭제하려고 하면 권한 오류가 발생합니다. 정책을 작성하는 관리자인 경우, 삭제 작업을 수행해야 하는 사용자에게 부여할 올바른 권한을 결정합니다.

사용자와 역할이 Amplify 콘솔을 계속 사용할 수 있도록 하려면 Amplify `ConsoleAccess` 또는 `ReadOnly` AWS 관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 에 대한 관리형 정책 AWS Amplify

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS 업데이트하는 AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 미칩니다. AWS AWS 서비스 는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하십시오.

AWS 관리형 정책: AdministratorAccess-Amplify

`AdministratorAccess-Amplify` 정책을 IAM 보안 인증에 연결할 수 있습니다. Amplify는 또한 이 정책을 서비스 역할에 연결하여 Amplify가 사용자를 대신하여 작업을 수행할 수 있습니다.

Amplify 콘솔에서 백엔드를 배포할 때는 Amplify가 AWS 리소스를 생성하고 관리하는 데 사용하는 `Amplify-Backend Deployment` 서비스 역할을 생성해야 합니다. IAM이 `AdministratorAccess-Amplify` 관리형 정책을 `Amplify-Backend Deployment` 서비스 역할에 연결합니다.

이 정책은 Amplify 애플리케이션이 백엔드를 생성하고 관리하는 데 필요한 리소스에 대한 직접 액세스를 명시적으로 허용하는 동시에 계정 관리 권한을 부여합니다.

권한 세부 정보

이 정책은 IAM 작업을 포함하여 여러 AWS 서비스에 대한 액세스를 제공합니다. 이러한 작업을 통해이 정책의 자격 증명을 AWS Identity and Access Management 사용하여 권한이 있는 다른 자격 증명을 생성할 수 있습니다. 이렇게 하면 권한 에스컬레이션이 가능하므로 이 정책은 `AdministratorAccess` 정책만큼 강력한 것으로 간주해야 합니다.

이 정책은 모든 리소스에 대한 `iam:PassRole` 작업 권한을 부여합니다. 이는 Amazon Cognito 사용자 풀 구성을 지원하는 데 필요합니다.

이 정책의 권한을 보려면 AWS 관리형 정책 참조에서 AdministratorAccess-Amplify를 참조하십시오.

AWS 관리형 정책: AmplifyBackendDeployFullAccess

`AmplifyBackendDeployFullAccess` 정책을 IAM 보안 인증에 연결할 수 있습니다.

이 정책은를 사용하여 Amplify 백엔드 리소스를 배포할 수 있는 전체 액세스 권한을 Amplify에 부여합니다 AWS 클라우드 개발 키트 (AWS CDK). 권한은 필요한 `AdministratorAccess` 정책 권한이 있는 AWS CDK 역할로 연기됩니다.

권한 세부 정보

이 정책에는 다음 작업을 수행할 수 있는 권한이 포함되어 있습니다.

- `Amplify`- 배포된 애플리케이션에 대한 메타데이터를 검색합니다.
- `AWS CloudFormation`– Amplify 관리형 스택을 생성, 업데이트, 삭제합니다.
- `SSM`- Amplify 관리형 SSM 파라미터 스토어 `String` 및 `SecureString` 파라미터를 생성, 업데이트, 삭제합니다.
- `AWS AppSync`- AWS AppSync 스키마, 해석기 및 함수 리소스를 업데이트하고 검색합니다. 목적은 Gen 2 샌드박스 핫스왑 기능을 지원하는 것입니다.
- `Lambda`- Amplify 관리형 함수의 구성을 업데이트하고 검색합니다. 목적은 Gen 2 샌드박스 핫스왑 기능을 지원하는 것입니다.

  Lambda 함수의 태그를 검색합니다. 목적은 고객이 정의한 Lambda 함수를 지원하는 것입니다.
- `Amazon S3`- Amplify 배포 자산을 검색합니다.
- `AWS Security Token Service`- AWS 클라우드 개발 키트 (AWS CDK) CLI가 배포 역할을 수임할 수 있도록 합니다.
- `Amazon RDS`– DB 인스턴스, 클러스터 및 프록시의 메타데이터를 읽습니다.
- `Amazon EC2`– 서브넷의 가용 영역 정보를 읽습니다.
- `CloudWatch Logs`- 고객의 Lambda 함수에 대한 로그를 검색합니다. 목적은 Amplify 클라우드 개발 샌드박스 환경이 Lambda 함수의 로그를 고객의 터미널로 스트리밍하도록 허용하는 것입니다.

이 정책의 권한을 보려면 AWS 관리형 정책 참조에서 AmplifyBackendDeployFullAccess를 참조하세요.

AWS 관리형 정책에 대한 Amplify 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 Amplify의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 에 대한 문서 기록 AWS Amplify 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AmplifyBackendDeployFullAccess — 기존 정책에 대한 업데이트	`logs:FilterLogEvents` 리소스에 읽기 액세스를 추가하여 Amplify가 사용자 지정 로그 그룹이 생성된 함수에서 로그를 스트리밍할 수 있도록 합	2024년 11월 14일

변경 사항	설명	날짜
	니다. 이는 Lambda 함수의 로그를 스트리밍하는 기존 기능의 확장입니다.	
AmplifyBackendDeployFullAccess — 기존 정책에 대한 업데이트	고객이 정의한 Lambda 함수를 지원하도록 `lambda:ListTags` 및 `logs:FilterLogEvents` 리소스에 대한 읽기 액세스를 추가합니다. 이러한 권한은 Amplify 클라우드 개발 샌드박스 환경이 Lambda 함수의 로그를 고객의 터미널로 스트리밍하도록 허용합니다.	2024년 7월 18일
AmplifyBackendDeployFullAccess — 기존 정책에 대한 업데이트	Amplify가 고객 계정에서 CDK 부트스트랩 버전을 감지할 수 있도록 `arn:aws:ssm:*:*:parameter/cdk-bootstrap/*` 리소스에 읽기 액세스를 추가합니다.	2024년 5월 31일

변경 사항	설명	날짜
AmplifyBackendDeployFullAccess — 기존 정책에 대한 업데이트	리소스 및 계정 조건 모두에 따라 범위가 지정되는 Amazon RDS 및 Amazon EC2 읽기 전용 권한이 포함된 새 `AmplifyDiscoverRDSVpcConfig` 정책문을 추가합니다. 이러한 권한은 고객이 기존 SQL 데이터베이스에서 Typescript 데이터 스키마를 생성할 수 있도록 허용하는 Amplify Gen 2 `npx amplify generate schema-from-database` 명령을 지원합니다. `rds:DescribeDBProxies` , `rds:DescribeDBInstances` , `rds:DescribeDBClusters` , `rds:DescribeDBSubnetGroups` 및 `ec2:DescribeSubnets` 권한을 추가합니다. 이 `npx amplify generate schema-from-database` 명령은 지정된 DB 호스트가 Amazon RDS에서 호스팅되는지 확인하고 SQL 데이터베이스에서 지원하는 AWS AppSync API를 설정하는 데 필요한 다른 리소스를 프로비저닝하는 데 필요한 Amazon VPC 구성을 자동 생성하는 데 이러한 권한이 필요합니다.	2024년 4월 17일

변경 사항	설명	날짜
AmplifyBackendDeployFullAccess — 기존 정책에 대한 업데이트	`DeleteBranch` API가 직접 호출될 때 스택 삭제를 지원하는 `cloudformation:DeleteStack` 정책 작업을 추가합니다. 핫스왑 기능을 지원하는 `lambda:GetFunction` 정책 작업을 추가합니다. Lambda 함수에 대한 업데이트를 지원하는 `lambda:UpdateFunctionConfiguration` 정책 작업을 추가합니다.	2024년 4월 5일
AdministratorAccess-Amplify – 기존 정책 업데이트	API에 대한 호출을 지원하는 `cloudformation:TagResource` 및 `cloudformation:UnTagResource` 권한을 추가합니다 AWS CloudFormation APIs.	2024년 4월 4일
AmplifyBackendDeployFullAccess — 기존 정책에 대한 업데이트	AWS 클라우드 개발 키트 (AWS CDK) 핫스왑을 지원하는 `lambda:InvokeFunction` 정책 작업을 추가합니다. AWS CDK 는 Lambda 함수를 직접 호출하여 Amazon S3 자산 핫스왑을 수행합니다. 핫스왑 기능을 지원하는 `lambda:UpdateFunctionCode` 정책 작업을 추가합니다.	2024년 1월 2일

변경 사항	설명	날짜
AmplifyBackendDeployFullAccess — 기존 정책에 대한 업데이트	`UpdateApiKey` 작업을 지원하는 정책 작업을 추가합니다. 리소스 삭제 없이 샌드박스를 종료하고 다시 시작한 후 앱을 성공적으로 배포할 수 있도록 하려면 이렇게 해야 합니다.	2023년 11월 17일
AmplifyBackendDeployFullAccess — 기존 정책에 대한 업데이트	Amplify 앱 배포를 지원하는 `amplify:GetBackendEnvironment` 권한을 추가합니다.	2023년 11월 6일
AmplifyBackendDeployFullAccess - 새 정책	Amplify가 Amplify 백엔드 리소스를 배포하는 데 필요한 최소 권한이 포함된 새 정책을 추가했습니다.	2023년 10월 8일
AdministratorAccess-Amplify – 기존 정책 업데이트	Amplify 명령줄 인터페이스(CLI)에 필요한 `ecr:DescribeRepositories` 권한을 추가합니다.	2023년 6월 1일

변경 사항	설명	날짜
AdministratorAccess-Amplify – 기존 정책 업데이트	AWS AppSync 리소스에서 태그 제거를 지원하는 정책 작업을 추가합니다. Amazon Polly 리소스를 지원하는 정책 작업을 추가합니다. OpenSearch 도메인 구성 업데이트를 지원하는 정책 작업을 추가합니다. AWS Identity and Access Management 역할에서 태그 제거를 지원하는 정책 작업을 추가합니다. Amazon DynamoDB 리소스에서 태그 제거를 지원하는 정책 작업을 추가합니다. Amplify 게시 및 호스팅 워크플로를 지원하도록 `cloudfront:GetCloudFrontOriginAccessIdentity` 및 `cloudfront:GetCloudFrontOriginAccessIdentityConfig` 권한을 `CLISDKCalls` 명령문 블록에 추가합니다. AWS CLI 이(가) 내부 버킷에 Amazon S3 Block Public Access를 활성화하는 Amazon S3 보안 모범 사례를 지원할 수 있도록 `s3:PutBucketPublicAccessBlo`	2023년 2월 24일

변경 사항	설명	날짜
	ck 권한을 CLIManage viaCFNPolicy 명령문 블록에 추가합니다. 명령CLISDKCalls 문 블록에 cloudformation:Des cribeStacks 권한을 추가하여 Amplify 백엔드 프로세서에서 재시도 시 고객의 AWS CloudFormation 스택 검색을 지원하여 스택이 업데이트되는 경우 실행이 중복되지 않도록 합니다. CLICloudformationP olicy 명령문 블록에 cloudformation:Lis tStacks 권한을 추가합니다. 이 권한은 CloudFormation DescriBESTacks 작업을 완벽하게 지원하는 데 필요합니다.	
AdministratorAccess-Amplify – 기존 정책 업데이트	Amplify 서버 측 렌더링 기능을 통해 애플리케이션 지표를 고객의 AWS 계정내 CloudWatch로 푸시할 수 있도록 정책 작업을 추가합니다.	2022년 8월 30일
AdministratorAccess-Amplify – 기존 정책 업데이트	Amplify 배포 Amazon S3 버킷에 대한 퍼블릭 액세스를 차단하는 정책 작업을 추가합니다.	2022년 4월 27일

변경 사항	설명	날짜
AdministratorAccess-Amplify – 기존 정책 업데이트	고객이 서버 측 렌더링(SSR) 앱을 삭제하도록 허용하는 작업을 추가합니다. 또한 이렇게 하면 해당 CloudFront 배포를 성공적으로 삭제할 수 있습니다. 고객이 Amplify CLI를 사용하여 기존 이벤트 소스의 이벤트를 처리할 수 있도록 다른 Lambda 함수를 지정하는 것을 허용하는 작업을 추가합니다. 이러한 변경으로 AWS Lambda 는 UpdateEventSourceMapping 작업을 수행할 수 있습니다.	2022년 4월 17일
AdministratorAccess-Amplify – 기존 정책 업데이트	모든 리소스에서 Amplify UI Builder 작업을 활성화하는 정책 작업을 추가합니다.	2021년 12월 2일
AdministratorAccess-Amplify – 기존 정책 업데이트	소셜 자격 증명 공급자를 사용하는 Amazon Cognito 인증 기능을 지원하는 정책 작업을 추가합니다. Lambda 계층을 지원하는 정책 작업을 추가합니다. Amplify 스토리지 범주를 지원하는 정책 작업을 추가합니다.	2021년 11월 8일

변경 사항	설명	날짜
AdministratorAccess-Amplify – 기존 정책 업데이트	Amplify Interactions 범주를 지원하는 Amazon Lex 작업을 추가합니다. Amplify Predictions 범주를 지원하는 Amazon Rekognition 작업을 추가합니다. Amazon Cognito 사용자 풀에서 MFA 구성을 지원하는 Amazon Cognito 작업을 추가합니다. CloudFormation 작업을 추가하여 AWS CloudFormation StackSets를 지원합니다. Amplify Geo 범주를 지원하는 Amazon Location Service 작업을 추가합니다. Amplify에서 Lambda 계층을 지원하는 Lambda 작업을 추가합니다. CloudWatch Events를 지원하는 CloudWatch Logs 작업을 추가합니다. Amplify Storage 범주를 지원하는 Amazon S3 작업을 추가합니다. 서버 측 렌더링(SSR) 앱을 지원하는 정책 작업을 추가합니다.	2021년 9월 27일

변경 사항	설명	날짜
AdministratorAccess-Amplify – 기존 정책 업데이트	모든 Amplify 작업을 단일 `amplify:*` 작업으로 통합합니다. 고객 Amazon S3 버킷의 암호화를 지원하는 Amazon S3 작업을 추가합니다. 권한 경계가 활성화된 Amplify 앱을 지원하도록 IAM 권한 경계 작업을 추가합니다. 발신 전화번호 보기, 대상 전화번호 보기, 생성, 확인, 삭제를 지원하는 Amazon SNS 작업을 추가합니다. Amplify Studio: Amplify 콘솔 및 Amplify Studio에서 백엔드를 관리할 수 있도록 Amazon Cognito, AWS Lambda IAM 및 AWS CloudFormation 정책 작업을 추가합니다. AWS Systems Manager (SSM) 정책 설명을 추가하여 Amplify 환경 암호를 관리합니다. Amplify 앱에 대한 Lambda 계층을 AWS CloudFormation `ListResources` 지원하는 작업을 추가합니다.	2021년 7월 28일

변경 사항	설명	날짜
Amplify의 변경 내용 추적 시작	Amplify는 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 7월 28일

Amplify 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Amplify 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amplify에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 AWS 계정 외부의 사람이 내 Amplify 리소스에 액세스하도록 허용하고 싶습니다.

Amplify에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `amplify:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 amplify:GetWidget on resource: my-example-widget
```

이 경우, `amplify:`*`GetWidget`* 작업을 사용하여 *`my-example-widget`* 리소스에 액세스할 수 있도록 `mateojackson` 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

Amplify Studio가 릴리스됨에 따라 앱 또는 백엔드를 삭제하려면 `amplify` 및 `amplifybackend` 권한이 모두 필요합니다. 관리자가 `amplify` 권한만 제공하는 IAM 정책을 작성한 경우, 앱을 삭제하려고 하면 권한 오류가 발생합니다.

다음 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`example-amplify-app`* 리소스를 삭제하려고 하지만 `amplifybackend:`*`RemoveAllBackends`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 amplifybackend;:RemoveAllBackends on resource: example-amplify-app
```

이 경우, Mateo는 *example-amplify-app* 작업을 사용하여 amplifybackend:*RemoveAllBackends* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amplify에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 Amplify에서 태스크를 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사람이 내 Amplify 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하십시오.

- Amplify에서 이러한 기능을 지원하는지 여부를 알아보려면 Amplify에서 IAM을 사용하는 방법 섹션을 참조하십시오.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.

- 리소스에 대한 액세스 권한을 타사에 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 소유에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

Amplify의 데이터 보호

AWS Amplify 는 AWS 공동 책임 모델 여기에는 데이터 보호에 대한 규정 및 지침이 포함됩니다. AWS 는 모든 AWS 서비스를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. AWS 는이 인프라에서 호스팅되는 데이터에 대한 제어를 유지합니다. 고객 콘텐츠 및 개인 데이터를 처리하기 위한 보안 구성 제어 포함. AWS 고객 및 APN 파트너, 데이터 컨트롤러 또는 데이터 프로세서로 작동 는 AWS 클라우드에 저장한 모든 개인 데이터에 대해 책임을 집니다.

데이터 보호를 위해 자격 AWS 계정 증명을 보호하고 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail.
- 서비스 내의 AWS 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.

이름 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마세요. 여기에는 Amplify 또는 기타 AWS 서비스에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. Amplify 또는 기타 서비스에 입력하는 모든 데이터를 진단 로그에 포함할 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시키지 마십시오.

데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

저장 시 암호화

유휴 데이터 암호화는 저장된 데이터를 암호화하여 무단 액세스로부터 데이터를 보호하는 것을 의미합니다. Amplify는 기본적으로에서 관리하는 Amazon S3 AWS KMS keys 용를 사용하여 앱의 빌드 아티팩트를 암호화합니다 AWS Key Management Service.

Amplify는 Amazon CloudFront를 사용하여 고객에게 앱을 서비스합니다. CloudFront는 엣지 로케이션 POP(상호 접속 위치)용으로 암호화된 SSD 및 Regional Edge Caches(REC)용으로 암호화된 EBS 볼륨을 사용합니다. CloudFront Functions의 함수 코드 및 구성은 항상 엣지 로케이션 POP의 암호화된 SSD와 CloudFront에서 사용하는 다른 스토리지 위치에 암호화된 형식으로 저장됩니다.

전송 중 암호화

전송 중 데이터 암호화는 데이터가 통신 엔드포인트 간을 이동하는 동안 데이터를 가로채기에서 보호하는 것을 의미합니다. Amplify Hosting은 전송 중 데이터에 대한 암호화를 기본으로 제공합니다. 고객과 Amplify 간 및 Amplify와 다운스트림 종속성 간의 모든 통신은 서명 버전 4 서명 프로세스를 사용하여 서명된 TLS 연결을 통해 보호됩니다. 모든 Amplify Hosting 엔드포인트는에서 관리하는 SHA-256 인증서를 사용합니다 AWS Private Certificate Authority. 자세한 내용은 서명 버전 4 서명 프로세스 및 정의 섹션을 참조하세요 AWS Private Certificate Authority.

암호화 키 관리

AWS Key Management Service (KMS)는 고객 데이터를 암호화하는 데 사용되는 AWS KMS keys암호화 키를 생성하고 제어하는 관리형 서비스입니다.는 고객을 대신하여 데이터를 암호화하기 위한 암호화 키를 AWS Amplify 생성하고 관리합니다. 관리할 암호화 키가 없습니다.

에 대한 규정 준수 검증 AWS Amplify

타사 감사자는 여러 규정 준수 프로그램의 AWS Amplify 일환으로의 보안 및 AWS 규정 준수를 평가합니다. 여기에는 SOC, PCI, ISO, HIPAA, MTCS, C5, K-ISMS, ENS High, OSPAR, HITRUST CSF, 및 FINMA가 포함됩니다.

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 제공 범위 섹션을 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다.는 규정 준수를 지원하기 위해 다음 리소스를 AWS 제공합니다.

- 보안 규정 준수 및 거버넌스 - 이러한 솔루션 구현 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수 기능을 배포하는 단계를 제공합니다.
- HIPAA 적격 서비스 참조 - HIPAA 적격 서비스가 나열되어 있습니다. 모두 HIPAA 자격이 AWS 서비스 있는 것은 아닙니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 - 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드에는 여러 프레임워크(미국 국립표준기술연구소(NIST), 결제 카드 산업 보안 표준 위원회(PCI), 국제표준화기구(ISO) 포함)의 보안 제어에 대한 지침을 보호하고 AWS 서비스 매핑하는 모범 사례가 요약되어 있습니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 -이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub - 내 보안 상태에 대한 포괄적인 보기를 AWS 서비스 제공합니다 AWS. Security Hub는 보안 컨트롤을 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하세요.
- Amazon GuardDuty - 의심스러운 악의적인 활동이 있는지 환경을 모니터링하여 사용자, AWS 계정 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty는 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 따르는 데 도움을 줄 수 있습니다.
- AWS Audit Manager - 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험과 규정 및 업계 표준 준수를 관리하는 방법을 간소화할 수 있습니다.

의 인프라 보안 AWS Amplify

관리형 서비스인는 AWS 글로벌 네트워크 보안으로 보호 AWS Amplify 됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 Amplify에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.

- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

Amplify의 보안 이벤트 로깅 및 모니터링

모니터링은 Amplify 및 다른 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 는 Amplify를 모니터링하고, 이상이 있을 때 보고하고, 필요한 경우 자동 조치를 취할 수 있도록 다음 모니터링 도구를 AWS 제공합니다.

- Amazon CloudWatch는 AWS 리소스와에서 실행하는 애플리케이션을 실시간으로 모니터링합니다 AWS. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 특정 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. Amplify를 통한 CloudWatch metrics 지표 및 경보 사용에 대한 자세한 내용은 Amplify 애플리케이션 모니터링을 참조하십시오.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, AWS CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.
- AWS CloudTrail는 AWS 계정에서 또는 계정을 대신하여 수행한 API 호출 및 관련 이벤트를 캡처하고 지정한 Amazon Simple Storage Service(Amazon S3) 버킷에 로그 파일을 전송합니다. 호출한 사용자 및 계정 AWS, 호출이 수행된 소스 IP 주소, 호출이 발생한 시기를 식별할 수 있습니다. 자세한 내용은 AWS CloudTrail를 사용하여 Amplify API 호출 로깅 단원을 참조하십시오.
- Amazon EventBridge: 애플리케이션을 다양한 소스의 데이터와 쉽게 연결할 수 있는 서버리스 이벤트 버스 서비스입니다. EventBridge는 자체 애플리케이션, 서비스형 소프트웨어(SaaS) 애플리케이션 및 AWS 서비스의 실시간 데이터 스트림을 제공한 다음, 해당 데이터를 AWS Lambda등의 대상으로 라우팅합니다. 이 방법을 통해 서비스에서 발생하는 이벤트를 모니터링하고 이벤트 기반 아키텍처를 구축할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서를 참조하십시오.

교차 서비스 혼동된 대리자 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 리소스에 다른 서비스를 AWS Amplify 제공하는 권한을 제한하는 것이 좋습니다. 두 전역 조건 컨텍스트 키를 모두 사용하는 경우 aws:SourceAccount 값과 aws:SourceArn 값의 계정은 동일한 정책 문에서 사용할 경우 동일한 계정 ID를 사용해야 합니다.

aws:SourceArn의 값은 Amplify 앱의 브랜치 ARN이어야 합니다. arn:*Partition*:amplify:*Region*:*Account*:apps/*AppId*/branches/*BranchName* 형식에서 이 값을 지정합니다.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모를 경우 또는 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 aws:SourceArn 전역 조건 컨텍스트 키를 사용합니다. 예제: arn:aws:*servicename*::*123456789012*:*.

다음 예는 계정의 Amplify 앱에 대한 액세스를 제한하고 혼동된 대리자 문제가 발생하는 것을 방지하기 위해 적용할 수 있는 역할 신뢰 정책을 보여줍니다. 이 정책을 사용하려면 정책 예제의 빨간색 기울임꼴 텍스트를 본인의 정보로 대체하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": [
          "amplify.me-south-1.amazonaws.com",
          "amplify.eu-south-1.amazonaws.com",
          "amplify.ap-east-1.amazonaws.com",
          "amplifybackend.amazonaws.com",
```

```
          "amplify.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

다음 예는 계정의 지정된 Amplify 앱에 대한 액세스를 제한하고 혼동된 대리자 문제가 발생하는 것을 방지하기 위해 적용할 수 있는 역할 신뢰 정책을 보여줍니다. 이 정책을 사용하려면 정책 예제의 빨간색 기울임꼴 텍스트를 본인의 정보로 대체하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": [
          "amplify.me-south-1.amazonaws.com",
          "amplify.eu-south-1.amazonaws.com",
          "amplify.ap-east-1.amazonaws.com",
          "amplifybackend.amazonaws.com",
          "amplify.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/d123456789/
branches/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
```

```
  }
}
```

Amplify의 보안 모범 사례

Amplify는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 지침이라기보다는 권장 사항으로만 사용해 주십시오.

Amplify 기본 도메인에 쿠키 사용

Amplify를 사용하여 웹 앱을 배포하는 경우, Amplify는 기본 `amplifyapp.com` 도메인에서 웹 앱을 대신 호스팅합니다. `https://branch-name.d1m7bkiki6tdw1.amplifyapp.com`와(과) 같은 형식의 URL에서 앱을 볼 수 있습니다.

Amplify 애플리케이션의 보안을 강화하기 위해 amplifyapp.com 도메인은 공개 접미사 목록(PSL)에 등록되어 있습니다. 보안 강화를 위해 Amplify 애플리케이션의 기본 도메인 이름에 민감한 쿠키를 설정해야 하는 경우, `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도(CSRF)로부터 도메인을 보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

Amplify Hosting 서비스 할당량

다음은 AWS Amplify 호스팅에 대한 서비스 할당량입니다. 서비스 할당량(과거에는 제한이라고도 함)은 AWS 계정의 최대 서비스 리소스 또는 작업 수입니다.

새로 AWS 계정 출시되어 앱 및 동시 작업 할당량이 줄었습니다.는 사용량에 따라 이러한 할당량을 자동으로 AWS 생성합니다. 할당량 증가 요청을 할 수도 있습니다.

Service Quotas 콘솔이 계정의 할당량에 관한 정보를 제공합니다. Service Quotas 콘솔을 사용하면 기본 할당량을 확인하고 조정 가능한 할당량에 대한 할당량 증가를 요청할 수 있습니다. 자세한 내용은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하십시오.

명칭	기본값	조정 가능	설명
앱	지원되는 각 리전: 25개	예	현재 리전의이 계정에서 AWS Amplify 콘솔에서 생성할 수 있는 최대 앱 수입니다.
앱당 브랜치 수	지원되는 각 리전: 50	아니요	현재 리전의 이 계정에서 생성할 수 있는 앱당 최대 브랜치 수.
빌드 아티팩트 크기	지원되는 각 리전: 5기가바이트	아니요	앱 빌드 아티팩트의 최대 크기(GB). 빌드 아티팩트는 빌드 후 AWS Amplify 콘솔에 의해 배포됩니다.
캐시 아티팩트 크기	지원되는 각 리전: 5기가바이트	아니요	캐시 아티팩트의 최대 크기(GB)입니다.
동시 작업	지원되는 각 리전: 5	예	현재 리전의 이 계정에서 생성할 수 있는 동시 작업의 최대 수입니다.

명칭	기본값	조정 가능	설명
앱당 도메인 수	지원되는 각 리전: 5	예	현재 리전의 이 계정에서 생성할 수 있는 앱당 도메인 최대 수.
환경 캐시 아티팩트 크기	지원되는 각 리전: 5기가바이트	아니요	환경 캐시 아티팩트의 최대 크기(GB)
수동 배포 ZIP 파일 크기	지원되는 각 리전: 5기가바이트	아니요	수동 배포 ZIP 파일의 최대 크기(GB).
시간당 최대 앱 생성 수	지원되는 각 리전: 25개	아니요	현재 리전의이 계정에서 시간당 AWS Amplify 콘솔에서 생성할 수 있는 최대 앱 수입니다.
초당 요청 토큰 수	지원되는 각 리전: 20개,000	예	앱의 초당 최대 요청 토큰 수입니다. Amplify Hosting은 요청이 사용하는 리소스(처리 시간 및 데이터 전송)의 양에 따라 요청에 토큰을 할당합니다.
도메인당 하위 도메인 수	지원되는 각 리전: 50	아니요	현재 리전의 이 계정에서 생성할 수 있는 도메인당 최대 하위 도메인 수.
앱당 웹훅 수	지원되는 각 리전: 50	예	현재 리전의 이 계정에서 생성할 수 있는 앱당 웹훅 최대 수.

Amplify 서비스 할당량에 대한 자세한 내용은 AWS 일반 참조의 AWS Amplify 엔드포인트 및 할당량을(를) 참조하십시오.

Amplify Hosting 문제 해결

Amplify Hosting 작업 시 문제가 발생한다면 이 섹션의 항목을 참조하세요.

주제

일반적인 Amplify 문제 해결

다음 정보는 Amplify Hosting과 관련된 문제를 해결하는 데 도움이 될 수 있습니다.

주제

HTTP 429 상태 코드(요청이 너무 많음)

Amplify는 수신 요청이 사용하는 처리 시간 및 데이터 전송을 기반으로 웹 사이트에 대한 초당 요청 수(RPS)를 제어합니다. 애플리케이션이 HTTP 429 상태 코드를 반환하면 수신 요청이 애플리케이션에 할당된 처리 시간 및 데이터 전송을 초과하는 것입니다. 이 애플리케이션 제한은 Amplify의 `REQUEST_TOKENS_PER_SECOND` 서비스 할당량에 의해 관리됩니다. 할당량에 대한 자세한 내용은 Amplify Hosting 서비스 할당량 섹션을 참조하세요.

이 문제를 해결하려면 애플리케이션을 최적화하여 요청 지속 시간을 줄이고 데이터 전송량을 줄여 앱의 RPS를 늘리는 것이 좋습니다. 예를 들어 토큰이 20,000개로 동일한 경우 100밀리초 이내에 응답하는 고도로 최적화된 SSR 페이지는 지연 시간이 200밀리초보다 긴 페이지와 비교하여 더 높은 RPS를 지원할 수 있습니다.

마찬가지로 1MB 응답 크기를 반환하는 애플리케이션은 250KB 응답 크기를 반환하는 애플리케이션보다 더 많은 토큰을 사용합니다.

또한 주어진 응답이 캐시에 유지되는 시간을 최대화하는 Cache-Control 헤더를 구성하여 Amazon CloudFront 캐시를 활용하는 것이 좋습니다. CloudFront 캐시에서 제공되는 요청은 속도 제한에 포함되지 않습니다. 각 CloudFront 배포는 초당 최대 250,000개의 요청을 처리할 수 있으므로 캐시를 사용하여 앱의 규모를 크게 조정할 수 있습니다. CloudFront 캐시에 대한 자세한 내용은 Amazon CloudFront 개발자 가이드의 캐싱 및 가용성 최적화를 참조하세요.

Amplify 콘솔에 앱의 빌드 상태 및 마지막 업데이트 시간이 표시되지 않음

Amplify 콘솔에서 모든 앱 페이지로 이동하면 현재 리전의 각 앱에 대한 타일이 표시됩니다. 배포됨 및 앱의 마지막 업데이트 시간과 같은 빌드 상태가 표시되지 않으면 앱에 연결된 `Production` 스테이지 브랜치가 없는 것입니다.

콘솔에서 앱을 나열하기 위해 Amplify는 `ListApps` API를 사용합니다. Amplify는 `ProductionBranch.status` 속성을 사용하여 빌드 상태를 표시하고 `ProductionBranch.lastDeployTime` 속성을 사용하여 마지막 업데이트 시간을 표시합니다. 이 API에 대한 자세한 내용은 Amplify Hosting API 설명서의 ProductionBranch를 참조하세요.

다음 지침에 따라 `Production` 단계를 앱의 브랜치에 연결합니다.

1. Amplify 콘솔에 로그인합니다.
2. 모든 앱 페이지에서 업데이트하려는 앱을 선택합니다.
3. 탐색 창에서 앱 설정을 선택한 다음 브랜치 설정을 선택합니다.
4. 브랜치 설정 섹션에서 편집을 선택합니다.
5. 프로덕션 브랜치에서 사용할 브랜치 이름을 선택합니다.
6. 저장을 선택합니다.
7. 모든 앱 페이지로 돌아갑니다. 이제 앱의 빌드 상태 및 마지막 업데이트 시간이 표시됩니다.

새 풀 요청에 대한 웹 미리 보기가 생성되지 않음

웹 미리 보기 기능을 사용하면 풀 요청의 변경 사항을 통합 브랜치에 병합하기 전에 미리 볼 수 있습니다. 웹 미리 보기는 리포지토리에 대한 모든 풀 요청을 기본 사이트에서 사용하는 URL과 다른 고유한 미리 보기 URL에 배포합니다.

앱에 대한 웹 미리 보기를 활성화했지만 새 PRs에 대해 생성되지 않는 경우 다음 중 하나가 문제의 원인인지 조사합니다.

1. 앱이 최대 `Branches per app` 서비스 할당량에 도달했는지 확인합니다. 할당량에 대한 자세한 내용은 Amplify Hosting 서비스 할당량 섹션을 참조하세요.

   앱당 브랜치 50개의 기본 할당량을 유지하려면 앱에서 자동 브랜치 삭제를 활성화하는 것이 좋습니다. 이렇게 하면 리포지토리에 더 이상 존재하지 않는 계정의 브랜치가 누적되지 않습니다.
2. 퍼블릭 GitHub 리포지토리를 사용하고 Amplify 앱에 IAM 서비스 역할이 연결되어 있는 경우 Amplify는 보안상의 이유로 미리 보기를 생성하지 않습니다. 예를 들어 백엔드가 있는 앱과 `WEB_COMPUTE` 호스팅 플랫폼에 배포된 앱에는 IAM 서비스 역할이 필요합니다. 따라서 리포지토리가 퍼블릭인 경우 이러한 유형의 앱에 대해 웹 미리 보기를 활성화할 수 없습니다.

   웹 미리 보기가 앱에서 작동하도록 하려면 서비스 역할의 연결을 해제하거나(앱에 백엔드가 없거나 `WEB_COMPUTE` 앱이 아닌 경우) GitHub 리포지토리를 비공개로 설정할 수 있습니다.

Amplify 콘솔에서 수동 배포가 보류 중 상태로 멈춤

수동 배포를 사용하면 Git 공급자를 연결하지 않고도 Amplify Hosting으로 웹 앱을 게시할 수 있습니다. 다음 네 가지 배포 옵션 중 하나를 사용할 수 있습니다.

1. Amplify 콘솔에서 애플리케이션 폴더를 끌어서 놓습니다.
2. Amplify 콘솔에서 .zip 파일(사이트의 빌드 아티팩트 포함)을 끌어서 놓습니다.
3. .zip 파일(사이트의 빌드 아티팩트 포함)을 Amazon S3 버킷에 업로드하고 Amplify 콘솔의 앱에 버킷을 연결합니다.
4. Amplify 콘솔에서 .zip 파일(사이트의 빌드 아티팩트 포함)을 가리키는 퍼블릭 URL을 사용합니다.

Amplify 콘솔에서 수동 배포에 애플리케이션 폴더를 사용할 때 끌어 놓기 기능과 관련된 문제를 알고 있습니다. 이러한 배포는 다음과 같은 이유로 실패할 수 있습니다.

- 일시적인 네트워크 문제가 발생합니다.

- 업로드 중에 파일에 로컬 변경 사항이 있습니다.
- 브라우저 세션은 대량의 정적 자산을 동시에 업로드하려고 시도합니다.

끌어서 놓기 업로드의 신뢰성을 개선하기 위해 노력하는 동안 애플리케이션 폴더를 끌어서 놓는 대신 .zip 파일을 사용하는 것이 좋습니다.

.zip 파일을 Amazon S3 버킷에 업로드하는 것이 좋습니다. 이렇게 하면 Amplify 콘솔에서 파일이 업로드되지 않고 수동 배포에 대한 안정성이 향상되기 때문입니다. Amplify와 Amazon S3의 통합은이 프로세스를 간소화합니다. 자세한 내용은 Amazon S3 버킷에서 Amplify에 정적 웹 사이트 배포 단원을 참조하십시오.

애플리케이션의 Node.js 버전을 업데이트해야 합니다.

Node.js 버전 16 및 18을 사용하는 앱에 대한 Amplify 지원은 2025년 9월 15일에 종료됩니다. 이미 배포된 애플리케이션은 계속 실행됩니다. 그러나이 날짜 이후에는 Node.js 버전 20 이상으로 업그레이드할 때까지 애플리케이션에 업데이트를 배포할 수 없습니다.

Amazon Linux 2023 빌드 이미지를 사용하는 경우 Node.js 버전 20이 기본적으로 지원됩니다. 2025년 9월 15일부터 AL2023 이미지는 자동으로 Node.js 22를 지원하고 기본 Node.js 버전을 18에서 22로 변경합니다.

Amazon Linux 2(AL2)는 Node.js 버전 20 이상을 자동으로 지원하지 않습니다. 현재 AL2를 사용하는 경우 AL2023으로 전환하는 것이 좋습니다. Amplify 콘솔에서 빌드 이미지를 변경할 수 있습니다. 지정한 Node.js 버전을 지원하는 사용자 지정 빌드 이미지를 사용할 수도 있습니다.

업그레이드하기 전에 새 브랜치에서 애플리케이션을 테스트하여 올바르게 작동하는지 확인하는 것이 좋습니다.

업그레이드 옵션

Amplify 콘솔

Amplify 콘솔에서 라이브 패키지 업데이트 기능을 사용하여 사용할 Node.js 버전을 지정할 수 있습니다. 지침은 빌드 이미지에서 특정 패키지 및 종속성 버전 사용 섹션을 참조하세요.

사용자 지정 빌드 이미지

사용자 지정 빌드 이미지를 사용 중이고 이미지에 NVM이 설치된 경우를 Dockerfile`nvm install 20`에 추가할 수 있습니다. 사용자 지정 빌드 이미지의 요구 사항 및 구성 지침에 대한 자세한 내용은 섹션을 참조하세요빌드 이미지 사용자 지정.

빌드 설정

명령을 preBuild `nvm use` 명령 섹션에 추가하여 애플리케이션의 `amplify.yml` 빌드 설정에 사용할 Node.js 버전을 지정할 수 있습니다. 애플리케이션의 빌드 설정 업데이트에 대한 지침은 섹션을 참조하세요Amplify 애플리케이션의 빌드 설정 구성.

다음 예제에서는 빌드 설정을 사용자 지정하여 라는 테스트 브랜치에서 기본 Node.js 버전을 Node.js 18로 설정하고 Node.js 버전 20으로 업그레이드하는 방법을 보여줍니다`node-20`.

```
frontend:
  phases:
    preBuild:
      commands:
        - nvm use 18
        - if [ "${AWS_BRANCH}" = "node-20" ]; then nvm use 20; fi
```

> **Warning**
>
> `preBuild` 명령은 라이브 패키지 업데이트 후 실행된다는 점에 유의하세요. `nvm use` 명령으로 지정된 Node.js 버전은 라이브 패키지 업데이트로 설정된 Node.js 버전을 재정의합니다.

Amazon Linux 2023 빌드 이미지 문제 해결

다음 정보는 Amazon Linux 2023(AL2023)와 관련된 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- Python 런타임으로 Amplify 함수를 실행하고 싶습니다.
- 슈퍼 사용자 또는 루트 권한이 필요한 명령을 실행하고 싶습니다.

Python 런타임으로 Amplify 함수를 실행하고 싶습니다.

이제 Amplify Hosting은 새 애플리케이션을 배포할 때 기본적으로 Amazon Linux 2023 빌드 이미지를 사용합니다. AL2023에는 Python 버전 3.8, 3.9, 3.10 및 3.11이 사전 설치되어 있습니다.

Amazon Linux 2 이미지와의 이전 버전 호환성을 위해 AL2023 빌드 이미지에는 사전 설치된 이전 버전의 Python에 대한 symlink가 있습니다.

기본적으로 Python 버전 3.10이 전 세계적으로 사용됩니다. 특정 Python 버전을 사용하여 함수를 빌드하려면 애플리케이션의 빌드 사양 파일에서 다음 명령을 실행합니다.

```
version: 1
backend:
  phases:
    build:
      commands:
        # use a python version globally
        - pyenv global 3.11
        # verify python version
        - python --version
        # install pipenv
        - pip install --user pipenv
        # add to path
        - export PATH=$PATH:/root/.local/bin
        # verify pipenv version
        - pipenv --version
        - amplifyPush --simple
```

슈퍼 사용자 또는 루트 권한이 필요한 명령을 실행하고 싶습니다.

Amazon Linux 2023 빌드 이미지를 사용하는 데 슈퍼 사용자 또는 루트 권한이 필요한 시스템 명령을 실행하면 오류가 발생하는 경우 Linux sudo 명령을 사용하여 해당 명령을 실행해야 합니다. 예를 들어, `yum install -y gcc` 실행 중 오류가 발생하면 `sudo yum install -y gcc`를 사용합니다.

Amazon Linux 2 빌드 이미지는 루트 사용자를 사용했지만 Amplify의 AL2023 이미지는 사용자 지정 `amplify` 사용자로 코드를 실행합니다. Amplify는 이 사용자에게 Linux sudo 명령을 사용하여 명령을 실행할 수 있는 권한을 부여합니다. 슈퍼 사용자 권한이 필요한 명령에 sudo를 사용하는 것이 모범 사례입니다.

빌드 문제 해결

Amplify 애플리케이션을 생성하거나 빌드할 때 문제가 발생하면이 섹션의 주제를 참조하여 도움을 받으세요.

주제

- 리포지토리에 대한 새 커밋이 Amplify 빌드를 트리거하지 않음
- 새 애플리케이션을 생성할 때 Amplify 콘솔에 리포지토리 이름이 나열되지 않음

- Cannot find module aws-exports 내 빌드가 오류와 함께 실패함(Gen 1 앱만 해당)
- 빌드 제한 시간을 재정의하고 싶습니다.

리포지토리에 대한 새 커밋이 Amplify 빌드를 트리거하지 않음

Git 리포지토리에 대한 새 커밋이 Amplify 빌드를 트리거하지 않는 경우 웹후크가 리포지토리에 여전히 있는지 확인합니다. 있는 경우 Webhook 요청 기록을 확인하여 장애가 있는지 확인합니다. Amplify는 수신 웹후크에 대해 페이로드 크기 제한이 256KB입니다. 변경된 파일이 많은 리포지토리에 커밋을 푸시하면이 제한을 초과하여 빌드가 트리거되지 않을 수 있습니다.

새 애플리케이션을 생성할 때 Amplify 콘솔에 리포지토리 이름이 나열되지 않음

Amplify 콘솔에서 새 애플리케이션을 생성할 때 리포지토리 및 브랜치 추가 페이지에서 조직의 사용 가능한 리포지토리 중에서 선택할 수 있습니다. 대상 리포지토리가 최근에 업데이트되지 않은 경우 목록에 표시되지 않을 수 있습니다. 이는 조직에 리포지토리가 많은 경우 발생할 수 있습니다. 이 문제를 해결하려면 커밋을 리포지토리로 푸시한 다음 콘솔에서 리포지토리 목록을 새로 고칩니다. 이렇게 하면 리포지토리가 표시되어야 합니다.

Cannot find module aws-exports 내 빌드가 오류와 함께 실패함(Gen 1 앱만 해당)

빌드 중에 앱이 `aws-exports.js` 파일을 찾을 수 없는 경우 다음 오류가 반환됩니다.

```
TS2307: Cannot find module 'aws-exports'
```

Amplify 명령줄 인터페이스(CLI)는 백엔드 빌드 중에 `aws-exports.js` 파일을 생성합니다. 이 오류를 해결하려면 빌드에 사용할 `aws-exports.js` 파일을 생성해야 합니다. 빌드 사양에 다음 코드를 추가하여 파일을 생성합니다.

```
backend:
  phases:
    build:
      commands:
        - "# Execute Amplify CLI with the helper script"
        - amplifyPush --simple
```

Amplify 앱에 대한 빌드 사양 설정의 전체 예는 섹션을 참조하세요빌드 사양 YAML 구문 참조.

빌드 제한 시간을 재정의하고 싶습니다.

기본 빌드 제한 시간은 30분입니다. _BUILD_TIMEOUT 환경 변수를 사용하여 기본 빌드 제한 시간을 재정의할 수 있습니다. 최소 빌드 제한 시간은 5분입니다. 최대 빌드 제한 시간은 120분입니다.

Amplify 콘솔에서 앱의 환경 변수를 설정하는 방법에 대한 지침은 섹션을 참조하세요환경 변수 설정.

사용자 지정 도메인 문제 해결

사용자 지정 도메인을 Amplify 애플리케이션에 연결할 때 문제가 발생하는 경우, 이 섹션의 항목을 참조하여 도움을 받으세요.

여기에서 문제 해결 방법을 찾을 수 없다면 지원에 문의하세요. 자세한 내용을 알아보려면 AWS Support 사용 설명서의 지원 사례 만들기를 참조하세요.

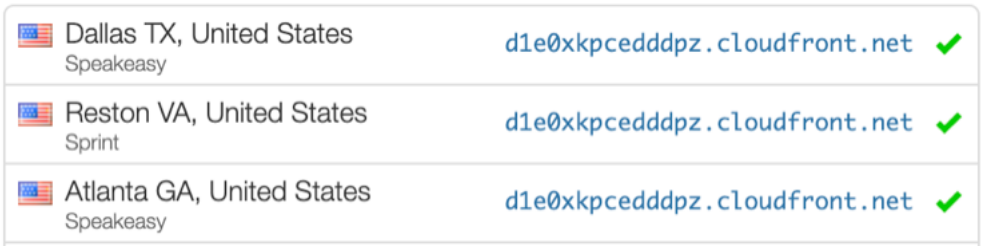
주제

- CNAME이 확인되는지 검증해야 합니다.
- 타사에서 호스팅하는 도메인이 확인 보류 상태로 고착된 경우,
- Amazon Route 53으로 호스팅된 도메인이 확인 보류 상태로 고착된 경우,
- 다중 수준 하위 도메인이 있는 앱이 확인 보류 중 상태에서 멈춤
- DNS 공급자가 정규화된 도메인 이름을 가진 A 레코드를 지원하지 않음
- CNAMEAlreadyExistsException 오류
- 추가 확인 필요 오류가 발생합니다.
- 클라우드프론트 URL에 404 오류가 발생합니다.
- 내 도메인을 방문할 때 SSL 인증서 또는 HTTPS 오류가 발생합니다.

CNAME이 확인되는지 검증해야 합니다.

1. 타사 도메인 공급자와 함께 DNS 레코드를 업데이트한 후에는 dig와 같은 도구나 https://www.whatsmydns.net/과 같은 무료 웹사이트를 사용하여 CNAME 레코드가 제대로 해결되고 있는지 확인할 수 있습니다. 다음 스크린샷은 whatsmydns.net을 사용하여 www.example.com 도메인의 CNAME 레코드를 확인하는 방법을 보여줍니다.

2. 검색을 선택하면 whatsmydns.net에 CNAME에 대한 결과가 표시됩니다. 다음 스크린샷은 CNAME이 cloudfront.net URL로 올바르게 확인되는지 확인하는 결과 목록의 예입니다.



타사에서 호스팅하는 도메인이 확인 보류 상태로 고착된 경우,

1. 커스텀 도메인이 확인 보류 중 상태에서 멈춘 경우, CNAME 레코드가 해결되고 있는지 확인하십시오. 이 작업을 수행하는 방법에 대한 지침은 이전 문제 해결 항목인 CNAME 문제가 해결되었는지 확인하는 방법을 참조하십시오.
2. CNAME 레코드가 확인되지 않는 경우, 도메인 공급자의 DNS 설정에 해당 CNAME 항목이 있는지 확인하십시오.

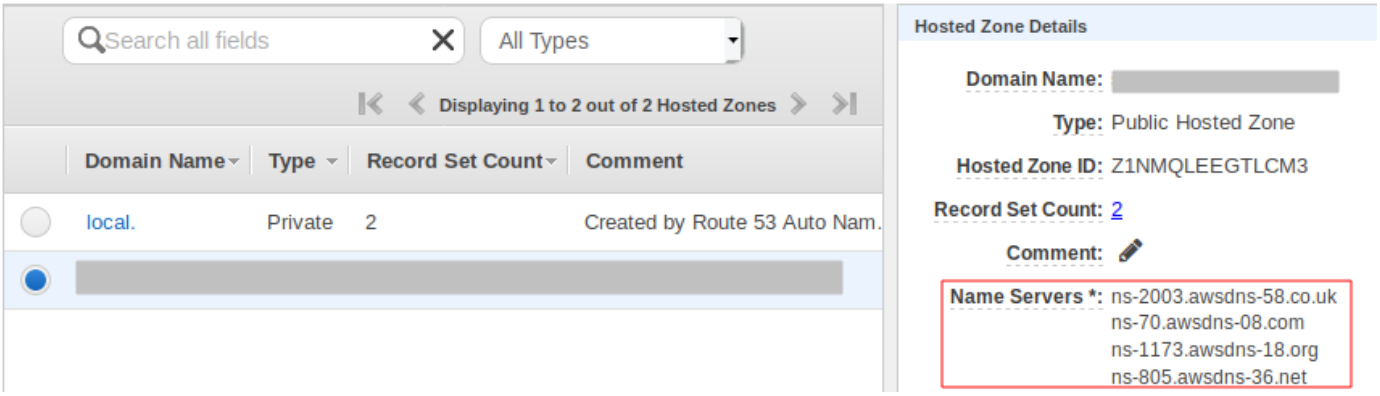
> **Important**
>
> 사용자 지정 도메인을 생성하는 즉시 CNAME 레코드를 업데이트하는 것이 중요합니다. 앱이 Amplify 콘솔에 생성되면 몇 분마다 CNAME 레코드의 리졸빙 여부를 확인해야 합니다. 1시간 후에 리졸빙이 되지 않는 경우, 몇 시간마다 확인을 해야 하므로 도메인의 사용 준비 완료에 지연이 초래될 수 있습니다. 앱을 만든 후 몇 시간 후에 CNAME 레코드를 추가하거나 업데이트한 경우, 이로 인해 앱이 확인 보류 중 상태에서 멈출 가능성이 높습니다.

3. CNAME 레코드가 존재하는지 확인한 경우, DNS 공급자에 문제가 있을 수 있습니다. DNS 공급자에게 연락하여 DNS 확인 CNAME 리졸빙이 되지 않는 이유를 진단하거나 DNS를 Route 53로 마이그레이션할 수 있습니다. 자세한 내용은 Amazon Route 53을 기존 도메인의 DNS 서비스로 지정 단원을 참조하십시오.

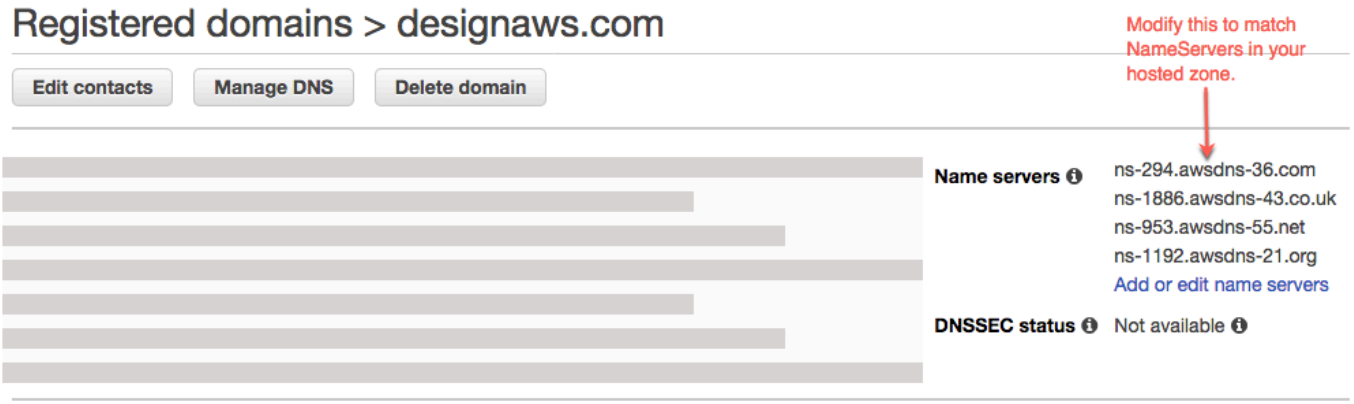
Amazon Route 53으로 호스팅된 도메인이 확인 보류 상태로 고착된 경우,

도메인을 Amazon Route 53으로 이전한 경우, 도메인은 앱 생성 시 Amplify에서 발급한 것과 다른 이름 서버를 가질 수 있습니다. 오류의 원인을 진단하려면 다음 단계를 수행하십시오.

1. Amazon Route 53 콘솔에 로그인합니다.
2. 탐색 창에서 호스팅 영역을 선택한 다음 연결 중인 도메인의 이름을 선택합니다.
3. 호스팅 영역 세부 정보 섹션에서 이름 서버 값을 기록합니다. 다음 단계를 완료하려면 이 값이 모두 필요합니다. Route 53 콘솔의 다음 스크린샷은 오른쪽 하단에 이름 서버 값의 위치를 표시합니다.



4. 탐색 창에서 등록된 도메인을 선택합니다. 등록된 도메인 섹션에 표시된 이름 서버가 호스팅 영역 세부 정보 섹션의 이전 단계에서 기록한 이름 서버 값과 일치하는지 확인하십시오. 일치하지 않는 경우, 이름 서버 값을 호스팅 영역의 값과 일치하도록 편집하십시오. Route 53 콘솔의 다음 스크린샷은 오른쪽에 이름 서버 값의 위치를 표시합니다.



5. 이렇게 해도 문제가 해결되지 않으면 지원에 문의하세요. 자세한 내용을 알아보려면 AWS Support 사용 설명서의 지원 사례 만들기를 참조하세요.

다중 수준 하위 도메인이 있는 앱이 확인 보류 중 상태에서 멈춤

타사 DNS 공급자에 연결할 때 다중 수준 하위 도메인이 있는 앱이 확인 보류 중 상태로 멈춘 경우 DNS 레코드 형식에 문제가 있을 수 있습니다. 일부 DNS 공급자는 2단계 도메인(SLD) 및 최상위 도메인(TLD) 도메인 접미사를 레코드에 자동으로 추가합니다. SLD 및 TLD가 포함된 형식으로 도메인을 지정하는 경우 도메인 확인 문제가 발생할 수 있습니다.

도메인을 연결할 때 먼저와 같이 Amplify에서 제공하는 전체 형식을 사용하여 도메인 이름을 지정해 보세요`_hash.docs.backend.example.com`. SSL 구성이 확인 보류 중 상태로 멈춘 경우 레코드에서 TLD 및 SLD를 제거해 보세요. 예를 들어 전체 형식이 인 경우를 `_hash.docs.backend.example.com`지정합니다`_hash.docs.backend`. 레코드가 전파될 때까지 15~30분 정도 기다립니다. 그런 다음 MX Toolbox와 같은 도구를 사용하여 확인 프로세스가 작동하는지 확인합니다.

DNS 공급자가 정규화된 도메인 이름을 가진 A 레코드를 지원하지 않음

일부 DNS 공급자는와 같이 FQDN(정규화된 도메인 이름)이 있는 A 레코드를 지원하지 않습니다`example.cloudfront.net`. 예를 들어 Cloudflare는 IPv4 주소만 쓸 A records 수 있으며를 지원하지 않습니다FQDNs. 이 제한을 해결하려면 DNS 구성A records에서 대신 CNAME 레코드를 사용하는 것이 좋습니다.

예를 들어 다음 DNS 구성은를 사용합니다A record.

```
A     | @ | ***.cloudfront.net
CNAME | www | ***.cloudfront.net
```

CNAME 레코드만 사용하도록 다음 DNS 구성으로 변경합니다.

```
CNAME | @ | ***.cloudfront.net
CNAME | www | ***.cloudfront.net
```

이 해결 방법을 사용하면 Cloudflare 시스템에서의 IPv4-only 제한을 방지하면서 정점 도메인(@ 레코드)을 A records CloudFront와 같은 서비스로 올바르게 지정할 수 있습니다.

CNAMEAlreadyExistsException 오류

CNAMEAlreadyExistsException 오류가 발생하는 경우, 연결하려는 호스트 이름 중 하나(하위 도메인이거나 정점 도메인)가 다른 Amazon CloudFront에 이미 배포되었음을 의미합니다. 오류의 원인은 현재 호스팅 및 DNS 공급자에 따라 다릅니다.

`example.com` 또는와 같은 CNAME별칭은 한 번에 단일 CloudFront 배포에만 연결할 `sub.example.com` 수 있습니다. CNAMEAlreadyExistsException은 도메인이 이미 동일한 내 또는 AWS 계정잠재적으로 다른 계정에 있는 다른 CloudFront 배포와 연결되어 있음을 나타냅니다. Amplify Hosting에서 생성한 새 배포가 작동하려면 먼저 도메인을 이전 CloudFront 배포와 연결 해제해야 합니다. 사용자 또는 조직이 여러을 소유한 경우 두 개 이상의 계정을 확인해야 AWS 계정할 수 있습니다.

다음 단계를 수행하여 CNAMEAlreadyExistsException 오류의 원인을 진단합니다.

1. Amazon CloudFront 콘솔에 로그인하고이 도메인이 다른 배포에 배포되지 않았는지 확인합니다. 한 번에 CNAME 레코드 하나만 CloudFront 배포에 연결할 수 있습니다.
2. 이전에 CloudFront 배포에 도메인을 배포한 경우, 해당 도메인을 제거해야 합니다.
   a. 왼쪽 탐색 창에서 배포를 선택합니다.
   b. 편집할 배포판 이름을 선택합니다.
   c. 일반 탭을 선택합니다. 설정 섹션에서 편집을 선택합니다.
   d. 대체 도메인 이름(CNAME)에서 도메인 이름을 제거합니다. 그런 다음 변경 사항 저장을 선택합니다.
3. 현재 AWS 계정 또는 다른에서이 도메인을 사용하는 다른 CloudFront 배포가 없는지 확인합니다 AWS 계정. 현재 실행 중인 서비스를 중단하지 않는 경우 호스팅 영역을 삭제하고 다시 생성해 보십시오.
4. 이 도메인이 사용자가 소유하는 다른 Amplify 앱에 연결되어 있는지 확인하십시오. 그렇다면 호스트 이름 중 하나를 재사용하려고 하지 마십시오. 다른 앱`www.example.com`에를 사용하는 경우 현재 연결 중인 앱과 `www.example.com` 함께를 사용할 수 없습니다. 와 같은 다른 하위 도메인을 사용할 수 있습니다`blog.example.com`.
5. 이 도메인이 다른 앱에 성공적으로 연결되었다가 최근 1시간 내에 삭제된 경우, 최소 1시간이 지난 후에 다시 시도하십시오. 6시간 후에도이 예외가 계속 표시되면에 문의하세요 지원. 자세한 내용을 알아보려면 AWS Support 사용 설명서의 지원 사례 만들기를 참조하세요.
6. Route 53을 통해 도메인을 관리하는 경우 이전 CloudFront 배포를 가리키는 호스팅 영역 CNAME 또는 ALIAS 레코드를 모두 정리해야 합니다.
7. 이전 단계를 완료한 후 Amplify Hosting에서 사용자 지정 도메인을 제거하고 워크플로를 다시 시작하여 Amplify 콘솔에서 사용자 지정 도메인을 연결합니다.

추가 확인 필요 오류가 발생합니다.

추가 확인 필요 오류가 발생하면 (AWS Certificate Manager ACM)에서이 인증서 요청을 처리하기 위해 추가 정보가 필요함을 의미합니다. 이는 예를 들어 도메인이 Alexa Top 1000 웹 사이트 순위에 든

경우, 사기 방지 조치로 이루어질 수 있습니다. 필요한 정보를 제공하려면 지원 센터를 사용하여 지원에 문의하십시오. 지원 계획이 없는 경우, ACM 토론 포럼에서 새 스레드를 게시할 수 있습니다.

> **Note**
>
> amazonaws.com, cloudfront.net 또는 elasticbeanstalk.com으로 끝나는 이름과 같이 Amazon 소유 도메인 이름에 대한 인증서를 요청할 수 없습니다.

클라우드프론트 URL에 404 오류가 발생합니다.

트래픽을 처리하기 위해 Amplify 호스팅은 CNAME 레코드를 통해 CloudFront URL을 가리킵니다. 앱을 사용자 지정 도메인에 연결하는 과정에서 Amplify 콘솔은 앱의 CloudFront URL을 표시합니다. 하지만 이 CloudFront URL을 사용하여 애플리케이션에 직접 액세스할 수는 없습니다. 404 오류가 반환됩니다. 애플리케이션은 Amplify 앱 URL(예: `https://main.d5udybEXAMPLE.amplifyapp.com`)또는 사용자 지정 도메인(예 `www.example.com`)을 사용해서만 리졸빙됩니다.

Amplify는 요청을 올바른 배포된 브랜치로 라우팅해야 하며 호스트 이름을 사용하여 이 작업을 수행합니다. 예를 들어 앱의 기본라인 브랜치를 가리키는 도메인 `www.example.com`을 구성할 수 있지만 동일한 앱의 dev 브랜치를 가리키도록 `dev.example.com`을 구성할 수도 있습니다. 따라서 Amplify가 요청을 적절하게 라우팅할 수 있도록 구성된 하위 도메인을 기반으로 애플리케이션을 방문해야 합니다.

내 도메인을 방문할 때 SSL 인증서 또는 HTTPS 오류가 발생합니다.

타사 DNS 공급자로 구성된 인증 기관 인증(CAA) DNS 레코드가 있는 경우 AWS Certificate Manager (ACM)이 사용자 지정 도메인 SSL 인증서에 대한 중간 인증서를 업데이트하거나 다시 발급하지 못할 수 있습니다. 이 문제를 해결하려면 Amazon 인증 기관 도메인 중 하나 이상을 신뢰하는 CAA 레코드를 추가해야 합니다. 다음 절차는 수행해야 하는 단계를 설명합니다.

Amazon 인증 기관을 신뢰할 수 있는 CAA 레코드를 추가하려면

1. Amazon 인증 기관 도메인 중 하나 이상을 신뢰하도록 도메인 공급자와 함께 CAA 레코드를 구성하십시오. CAA 레코드 구성에 대한 자세한 내용은AWS Certificate Manager 사용 설명서의 CAA(인증 기관 승인) 문제를 참조하십시오.
2. 다음 방법 중 하나를 사용하여 SSL 인증서를 업데이트합니다.
   - Amplify 콘솔을 사용하여 수동으로 업데이트합니다.

> **Note**
>
> 이렇게 하면 사용자 지정 도메인이 다운될 수 있습니다.

a. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
b. CAA 레코드를 추가할 앱을 선택합니다.
c. 탐색 창에서 앱 설정, 도메인 관리를 선택합니다.
d. 도메인 관리 페이지에서 커스텀 도메인을 삭제합니다.
e. 앱을 사용자 지정 도메인에 다시 연결합니다. 이 프로세스는 새 SSL 인증서를 발급하며, 이제 ACM에서 해당 중간 인증서를 관리할 수 있습니다.

   앱을 사용자 지정 도메인에 다시 연결하려면 사용 중인 도메인 공급자에 해당하는 다음 절차 중 하나를 사용하십시오.

   - Amazon Route 53에서 관리하는 사용자 지정 도메인 추가.
   - 타사 DNS 공급자가 관리하는 사용자 지정 도메인 추가.
   - GoDaddy에서 관리하는 도메인의 DNS 레코드 업데이트.

- SSL 인증서를 다시 발급 지원 하려면에 문의하세요.

서버 측 렌더링 애플리케이션 문제 해결

Amplify Hosting 컴퓨팅을 사용하여 SSR 앱을 배포할 때 예상치 못한 문제가 발생하는 경우, 다음 문제 해결 항목을 검토하십시오. 여기에서 문제 해결 방법을 찾을 수 없는 경우, Amplify Hosting GitHub Issues 리포지토리의 SSR 웹 컴퓨팅 문제 해결 가이드를 참조하십시오.

주제

- 프레임워크 어댑터를 사용하는 데 도움이 필요
- 엣지 API 경로로 인해 Next.js 빌드가 실패함
- 앱에 온디맨드 증분 정적 재생성이 작동하지 않음
- 애플리케이션의 빌드 출력이 허용되는 최대 크기를 초과함
- 메모리 부족 오류로 인해 빌드가 실패함
- 애플리케이션의 HTTP 응답 크기가 너무 큼
- 컴퓨팅 앱의 시작 시간을 로컬에서 측정하려면 어떻게 해야 하나요?

프레임워크 어댑터를 사용하는 데 도움이 필요

프레임워크 어댑터를 사용하는 SSR 앱 배포에서 문제가 발생하면 모든 SSR 프레임워크에 오픈 소스 어댑터 사용 섹션을 참조하세요.

엣지 API 경로로 인해 Next.js 빌드가 실패함

현재 Amplify는 Next.js Edge API 라우팅을 지원하지 않습니다. Amplify로 앱을 호스팅할 때는 비엣지 API와 미들웨어를 사용해야 합니다.

앱에 온디맨드 증분 정적 재생성이 작동하지 않음

Next.js 버전 12.2.0부터 ISR(증분 정적 재생성)을 지원하여 특정 페이지의 Next.js 캐시를 수동으로 제거합니다. 그러나 Amplify는 현재 온디맨드 ISR을 지원하지 않습니다. 앱이 Next.js 온디맨드 재검증을 사용하는 경우, 앱을 Amplify에 배포하면 이 기능이 작동하지 않습니다.

애플리케이션의 빌드 출력이 허용되는 최대 크기를 초과함

현재 Amplify가 SSR 앱에 지원하는 최대 빌드 출력 크기는 220MB입니다. 앱의 빌드 출력 크기가 허용되는 최대 크기를 초과한다는 오류 메시지가 표시되면 이를 줄이기 위한 조치를 취해야 합니다.

앱의 빌드 출력 크기를 줄이려면 앱의 빌드 아티팩트를 검사하여 업데이트하거나 제거할 대규모 종속성을 식별할 수 있습니다. 먼저 빌드 아티팩트를 로컬 컴퓨터에 다운로드합니다. 그런 다음 디렉터리의 크기를 확인합니다. 예를 들어 `node_modules` 디렉터리에는 Next.js 서버 런타임 파일에서 참조하는 `@swc` 및 `@esbuild` 같은 바이너리가 포함될 수 있습니다. 런타임에는 이러한 바이너리가 필요하지 않으므로 빌드 후 삭제할 수 있습니다.

다음 지침에 따라 앱의 빌드 출력을 다운로드하고 AWS Command Line Interface (CLI)를 사용하여 디렉터리 크기를 검사합니다.

Next.js 앱의 빌드 출력을 다운로드하고 검사하려면

1. 터미널 창을 열고 다음 명령을 실행합니다. 앱 ID, 브랜치 이름 및 작업 ID를 자신의 정보로 변경합니다. 작업 ID에는 조사 중인 실패한 빌드의 빌드 번호를 사용합니다.

   ```
   aws amplify get-job --app-id abcd1234 --branch-name main --job-id 2
   ```

2. 터미널 출력의 `job`, `steps`, `stepName: "BUILD"` 섹션에서 미리 서명된 아티팩트 URL을 찾습니다. URL은 다음 예제 출력에서 빨간색으로 강조 표시되어 있습니다.

```
"job": {
    "summary": {
        "jobArn": "arn:aws:amplify:us-west-2:111122223333:apps/abcd1234/main/
jobs/0000000002",
        "jobId": "2",
        "commitId": "HEAD",
        "commitTime": "2024-02-08T21:54:42.398000+00:00",
        "startTime": "2024-02-08T21:54:42.674000+00:00",
        "status": "SUCCEED",
        "endTime": "2024-02-08T22:03:58.071000+00:00"
    },
    "steps": [
        {
            "stepName": "BUILD",
            "startTime": "2024-02-08T21:54:42.693000+00:00",
            "status": "SUCCEED",
            "endTime": "2024-02-08T22:03:30.897000+00:00",
            "logUrl": "https://aws-amplify-prod-us-west-2-artifacts.s3.us-
west-2.amazonaws.com/abcd1234/main/0000000002/BUILD/log.txt?X-Amz-Security-
Token=IQoJb3JpZ2luX2V...Example
```

3. URL을 복사하여 브라우저 창에 붙여넣습니다. `artifacts.zip` 파일이 로컬 컴퓨터에 다운로드됩니다. 이 파일이 빌드 출력입니다.
4. du 디스크 사용량 명령을 실행하여 디렉터리의 크기를 검사합니다. 다음 예제 명령은 `compute` 및 `static` 디렉터리의 크기를 반환합니다.

```
du -csh compute static
```

다음은 `compute` 및 `static` 디렉터리의 크기 정보가 포함된 출력의 예입니다.

```
 29M    compute
3.8M    static
 33M    total
```

5. `compute` 디렉터리를 열고 `node_modules` 폴더를 찾습니다. 폴더 크기를 줄이기 위해 업데이트하거나 제거할 수 있는 파일의 종속성을 검토합니다.
6. 앱에 런타임에 필요하지 않은 바이너리가 포함된 경우 빌드 후 앱의 `amplify.yml` 파일에서 빌드 섹션에 다음 명령을 추가하여 해당 바이너리를 삭제합니다.

```
- rm -f node_modules/@swc/core-linux-x64-gnu/swc.linux-x64-gnu.node
```

```
- rm -f node_modules/@swc/core-linux-x64-musl/swc.linux-x64-musl.node
```

다음은 프로덕션 빌드를 실행한 후 이러한 명령이 추가된 amplify.yml 파일의 빌드 명령 섹션의 예입니다.

```
frontend:
  phases:
    build:
      commands:
         -npm run build

         // After running a production build, delete the files
         - rm -f node_modules/@swc/core-linux-x64-gnu/swc.linux-x64-gnu.node
         - rm -f node_modules/@swc/core-linux-x64-musl/swc.linux-x64-musl.node
```

메모리 부족 오류로 인해 빌드가 실패함

Next.js를 통해 빌드 아티팩트를 캐시하여 후속 빌드의 성능을 개선할 수 있습니다. 또한 Amplify의 AWS CodeBuild 컨테이너는 사용자를 대신하여이 캐시를 압축하고 Amazon S3에 업로드하여 후속 빌드 성능을 개선합니다. 이로 인해 메모리 부족 오류가 발생하면 빌드가 실패할 수 있습니다.

빌드 단계에서 앱이 메모리 제한을 초과하지 않도록 하려면 다음 작업을 수행합니다. 먼저 빌드 설정의 cache.paths 섹션에서 `.next/cache/**/*`를 삭제합니다. 그런 다음, 빌드 설정 파일에서 `NODE_OPTIONS` 환경 변수를 제거합니다. 대신 Amplify 콘솔에서 `NODE_OPTIONS` 환경 변수를 설정하여 노드 최대 메모리 제한을 정의합니다. Amplify 콘솔을 사용하여 환경 변수를 설정하는 방법에 대한 자세한 내용은 환경 변수 설정을 참조하십시오.

이러한 변경을 수행한 후 빌드를 다시 시도합니다. 빌드가 성공하면 빌드 설정 파일의 cache.paths 섹션에 `.next/cache/**/*`를 다시 추가합니다.

빌드 성능을 개선하기 위한 Next.js 캐시 구성에 대한 자세한 내용은 Next.js 웹사이트의 AWS CodeBuild를 참조하십시오.

애플리케이션의 HTTP 응답 크기가 너무 큼

현재 Amplify가 웹 컴퓨팅 플랫폼을 사용하는 Next.js 12 이상 앱에 지원하는 최대 응답 크기는 5.72MB입니다. 이 한도를 초과하는 응답은 콘텐츠가 없는 504 오류를 클라이언트에 반환합니다.

컴퓨팅 앱의 시작 시간을 로컬에서 측정하려면 어떻게 해야 하나요?

다음 지침에 따라 Next.js 12 이상 컴퓨팅 앱의 로컬 초기화/시작 시간을 결정합니다. 앱의 성능을 로컬에서 Amplify Hosting과 비교하고 결과를 사용하여 앱의 성능을 개선할 수 있습니다.

Next.js 컴퓨팅 앱의 초기화 시간을 로컬에서 측정하려면

1. 앱의 `next.config.js` 파일을 열고 다음과 `standalone` 같이 `output` 옵션을 로 설정합니다.

```
** @type {import('next').NextConfig} */
const nextConfig = {
  // Other options
  output: "standalone",
};

module.exports = nextConfig;
```

2. 터미널 창을 열고 다음 명령을 실행하여 앱을 빌드합니다.

```
next build
```

3. 다음 명령을 실행하여 `.next/static` 폴더를에 복사합니다`.next/standalone/.next/static`.

```
cp -r .next/static .next/standalone/.next/static
```

4. 다음 명령을 실행하여 `public` 폴더를에 복사합니다`.next/standalone/public`.

```
cp -r public .next/standalone/public
```

5. 다음 명령을 실행하여 Next.js 서버를 시작합니다.

```
node .next/standalone/server.js
```

6. 5단계에서 명령을 실행하고 서버를 시작하는 데 걸리는 시간을 기록해 둡니다. 서버가 포트에서 수신 대기 중인 경우 다음 메시지를 인쇄해야 합니다.

```
Listening on port 3000
```

7. 6단계에서 서버를 시작한 후 다른 모듈을 로드하는 데 걸리는 시간을 기록해 둡니다. 예를 들어 같은 라이브러리를 로드하는 데 10~12초가 bugsnag 걸립니다. 로드되면 확인 메시지가 표시됩니다[bugsnag] loaded.
8. 6단계와 7단계의 기간을 함께 추가합니다. 이 결과는 컴퓨팅 앱의 로컬 초기화/시작 시간입니다.

리디렉션 및 재작성 문제 해결

Amplify 애플리케이션에 대한 리디렉션 및 재작성을 설정할 때 문제가 발생하는 경우이 섹션의 주제를 참조하여 도움을 받으세요.

주제

- SPA 리디렉션 규칙을 사용하더라도 특정 경로에 대한 액세스가 거부됩니다.
- API에 대한 역방향 프록시를 설정하려고 합니다.

SPA 리디렉션 규칙을 사용하더라도 특정 경로에 대한 액세스가 거부됩니다.

SPA 리디렉션 규칙이 있는 특정 경로에 대해 액세스 거부 오류가 발생하는 경우 앱의 빌드 설정에서가 올바르게 설정되지 않을 baseDirectory 수 있습니다. 예를 들어 앱의 프런트엔드가 build 디렉터리에 빌드된 경우 빌드 설정도 build 디렉터리를 가리켜야 합니다. 다음 빌드 사양 예제에서는이 설정을 보여줍니다.

```
frontend:
  artifacts:
    baseDirectory: build
    files:
        - "**/*"
```

Amplify 앱에 대한 빌드 사양 설정의 전체 예는 섹션을 참조하세요. 빌드 사양 YAML 구문 참조

API에 대한 역방향 프록시를 설정하려고 합니다.

다음 JSON을 사용하여 동적 엔드포인트에 대한 역방향 프록시를 설정할 수 있습니다.

```
[
  {
    "source": "/documents/<*>",
    "target": "https://otherdomain/resource/<*>",
```

```
    "status": "200",
    "condition": null
  }
]
```

Amplify 앱을 타사 API로 역방향 프록시를 생성하는 기본 예제는 섹션을 참조하세요역방향 프록시 다시 쓰기.

캐싱 문제 해결

Amplify 애플리케이션에 캐싱 문제가 발생하는 경우이 섹션의 주제를 참조하여 도움을 받으세요.

주제

- 앱의 캐시 크기를 줄이고 싶습니다.
- 앱의 캐시에서 읽기를 비활성화하고 싶습니다.

앱의 캐시 크기를 줄이고 싶습니다.

캐시를 사용하는 경우 빌드 간에 정리되지 않은 중간 파일을 캐싱할 수 있습니다. 자주 사용하지 않는 파일을 캐싱하면 캐시 크기가 증가합니다. 이를 방지하려면 앱 빌드 사양의 cache 섹션에 있는 ! 명령을 사용하여 특정 폴더가 캐시되지 않도록 제외할 수 있습니다.

다음 빌드 설정 예제에서는 ! 명령을 사용하여 캐시하지 않을 폴더를 지정하는 방법을 보여줍니다.

```
cache:
  paths:
    - node_modules/**/*
    - "!node_modules/path/not/to/cache"
```

node_modules 폴더를 캐싱하면 node_modules/.cache가 기본적으로 생략됩니다.

Amplify 앱에 대한 빌드 사양 설정의 전체 예는 섹션을 참조하세요. 빌드 사양 YAML 구문 참조

앱의 캐시에서 읽기를 비활성화하고 싶습니다.

앱의 캐시에서 읽기를 비활성화하려면 앱의 빌드 사양에서 캐시 섹션을 제거합니다.

GitHub 리포지토리에 대한 Amplify 액세스 설정

Amplify는 이제 GitHub 앱 기능을 사용하여 GitHub 리포지토리에 대한 Amplify의 읽기 전용 액세스를 승인합니다. Amplify GitHub 앱을 사용하면 권한이 더욱 세밀하게 조정되므로 지정한 리포지토리에만 Amplify에 액세스 권한을 부여할 수 있습니다. GitHub 앱에 대해 자세히 알아보려면 GitHub 웹 사이트의 GitHub 앱 정보를 참조하십시오.

GitHub 리포지토리에 저장된 새 앱을 연결하면 Amplify는 기본적으로 GitHub 앱을 사용하여 리포지토리에 액세스합니다. 하지만 이전에 GitHub 리포지토리에서 연결한 기존 Amplify 앱은 액세스에 OAuth를 사용합니다. CI/CD는 이러한 앱에서 계속 작동하지만 새로운 Amplify GitHub 앱을 사용하도록 마이그레이션하는 것이 좋습니다.

Amplify 콘솔을 사용하여 새 앱을 배포하거나 기존 앱을 마이그레이션하면 Amplify GitHub 앱의 설치 위치로 자동 안내됩니다. 앱의 설치 랜딩 페이지에 수동으로 액세스하려면 웹 브라우저를 열고 리전별로 앱을 탐색하십시오. 형식 `https://github.com/apps/aws-amplify-REGION`을(를) 사용하여 ##을 Amplify 앱을 배포할 리전으로 대체하십시오. 예를 들어, 미국 서부(오레곤) 리전에 Amplify GitHub 앱을 설치하려면 https://github.com/apps/aws-amplify-us-west-2으로 이동하십시오.

주제

- 새 배포를 위한 Amplify GitHub 앱 설치 및 권한 부여
- 기존 OAuth 앱을 Amplify GitHub 앱으로 마이그레이션하기
- AWS CloudFormation, CLI 및 SDK 배포를 위한 Amplify GitHub 앱 설정
- Amplify GitHub 앱을 사용하여 웹 미리 보기 설정하기

새 배포를 위한 Amplify GitHub 앱 설치 및 권한 부여

GitHub 리포지토리의 기존 코드를 사용하여 Amplify에 새 앱을 배포하는 경우, 다음 지침에 따라 GitHub 앱을 설치하고 승인하십시오.

Amplify GitHub 앱을 설치하고 승인하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 모든 앱 페이지에서 새 앱을 선택한 다음 웹 앱 호스팅을 선택합니다.
3. Amplify Hosting 시작하기 페이지에서 GitHub를 선택한 다음 계속을 선택합니다.
4. GitHub 리포지토리를 처음 연결하는 경우 GitHub.com,의 브라우저에 새 페이지가 열리고 GitHub 계정 AWS Amplify 에서 권한 부여 권한을 요청합니다. Authorize를 선택합니다.

5. 다음으로 GitHub 계정에 Amplify GitHub 앱을 설치해야 합니다. Github.com GitHub 계정에를 설치하고 권한을 부여할 수 AWS Amplify 있는 권한을 요청하는 페이지가 열립니다.
6. Amplify GitHub 앱을 설치할 GitHub 계정을 선택합니다.
7. 다음 중 하나를 수행하십시오.
   - 모든 리포지토리에 설치를 적용하려면 모든 리포지토리를 선택합니다.
   - 선택한 특정 리포지토리로 설치를 제한하려면 리포지토리만 선택을 선택합니다. 선택한 리포지토리에 마이그레이션하려는 앱의 리포지토리를 포함해야 합니다.
8. 설치 및 승인을 선택합니다.
9. Amplify 콘솔에서 앱의 리포지토리 분기 추가 페이지로 리디렉션됩니다.
10. 최근 업데이트된 리포지토리 목록에서 연결할 리포지토리 이름을 선택합니다.
11. 브랜치 목록에서 연결할 리포지토리 브랜치의 이름을 선택합니다.
12. 다음을 선택합니다.
13. 보안 설정 구성 페이지에서 다음을 선택합니다.
14. 검토 페이지에서 저장 및 배포를 선택합니다.

기존 OAuth 앱을 Amplify GitHub 앱으로 마이그레이션하기

이전에 GitHub 리포지토리에서 연결한 기존 Amplify 앱은 리포지토리 액세스에 OAuth를 사용합니다. Amplify GitHub 앱을 사용하려면 이러한 앱을 마이그레이션하는 것이 좋습니다.

다음 지침에 따라 앱을 마이그레이션하고 GitHub 계정에서 해당 OAuth 웹후크를 삭제하십시오. 참고로 마이그레이션 절차는 Amplify GitHub 앱이 이미 설치되어 있는지 여부에 따라 달라집니다. 첫 번째 앱을 마이그레이션하고 GitHub 앱을 설치 및 승인한 후에는 후속 앱 마이그레이션을 위해 리포지토리 권한만 업데이트하면 됩니다.

앱을 OAuth에서 GitHub 앱으로 마이그레이션하려면

1. 에 로그인 AWS Management Console 하고 Amplify 콘솔을 엽니다.
2. 마이그레이션할 앱을 선택합니다.
3. 앱의 정보 페이지에서 파란색 GitHub 앱으로 마이그레이션 메시지를 찾아 마이그레이션 시작을 선택합니다.
4. GitHub 앱 설치 및 승인 페이지에서 GitHub 앱 구성을 선택합니다.
5. GitHub.com의 브라우저에 GitHub 계정에서 AWS Amplify 를 승인할 수 있는 권한을 요청하는 새 페이지가 열립니다. Authorize를 선택합니다.

6. Amplify GitHub 앱을 설치할 GitHub 계정을 선택합니다.
7. 다음 중 하나를 수행하십시오.
   - 모든 리포지토리에 설치를 적용하려면 모든 리포지토리를 선택합니다.
   - 선택한 특정 리포지토리로 설치를 제한하려면 리포지토리만 선택을 선택합니다. 선택한 리포지토리에 마이그레이션하려는 앱의 리포지토리를 포함해야 합니다.
8. 설치 및 승인을 선택합니다.
9. Amplify 콘솔의 앱에 대한 GitHub 앱 설치 및 승인 페이지로 리디렉션됩니다. GitHub 인증에 성공하면 성공 메시지가 표시됩니다. 다음을 선택합니다.
10. 설치 완료 페이지에서 설치 완료를 선택합니다. 이 단계는 기존 웹후크를 삭제하고, 새 웹후크를 만들고, 마이그레이션을 완료합니다.

AWS CloudFormation, CLI 및 SDK 배포를 위한 Amplify GitHub 앱 설정

이전에 GitHub 리포지토리에서 연결한 기존 Amplify 앱은 리포지토리 액세스에 OAuth를 사용합니다. 여기에는 Amplify 명령줄 인터페이스(CLI) AWS CloudFormation또는 SDKs. 새 Amplify GitHub 앱을 사용하려면 이러한 앱을 마이그레이션하는 것이 좋습니다. 마이그레이션은 Amplify 콘솔의 AWS Management Console에서 수행해야 합니다. 지침은 기존 OAuth 앱을 Amplify GitHub 앱으로 마이그레이션하기 섹션을 참조하세요.

AWS CloudFormation Amplify CLI 및 SDKs를 사용하여 리포지토리 액세스를 위해 GitHub 앱을 사용하는 새 Amplify 앱을 배포할 수 있습니다. 이 프로세스를 진행하려면 먼저 GitHub 계정에 Amplify GitHub 앱을 설치해야 합니다. 다음으로 GitHub 계정에서 개인용 액세스 토큰을 생성해야 합니다. 마지막으로 앱을 배포하고 개인용 액세스 토큰을 지정합니다.

계정에 Amplify GitHub 앱을 설치하십시오.

1. 웹 브라우저를 열고 앱을 배포할 AWS 리전에서 Amplify GitHub 앱의 설치 위치로 이동합니다.

   *##*을 사용자가 직접 입력한 것으로 대체하는 형식 `https://github.com/apps/aws-amplify-`*`REGION`*`/installations/new`을(를) 사용하십시오. 예를 들어 미국 서부(오레곤) 리전에 앱을 설치하는 경우, `https://github.com/apps/aws-amplify-us-west-2/installations/new`을(를) 지정합니다.
2. Amplify GitHub 앱을 설치할 GitHub 계정을 선택합니다.
3. 다음 중 하나를 수행하십시오.
   - 모든 리포지토리에 설치를 적용하려면 모든 리포지토리를 선택합니다.

- 선택한 특정 리포지토리로 설치를 제한하려면 리포지토리만 선택을 선택합니다. 선택한 리포지토리에 마이그레이션하려는 앱의 리포지토리를 포함해야 합니다.

4. 설치를 선택합니다.

GitHub 계정에서 개인용 액세스 토큰 생성

1. GitHub 계정에 가입합니다.
2. 오른쪽 상단에서 프로필 사진을 찾아 메뉴에서 설정을 선택합니다.
3. 왼쪽 탐색 메뉴에서 개발자 설정을 선택합니다.
4. GitHub 앱 페이지의 왼쪽 탐색 메뉴에서 개인용 액세스 토큰을 선택합니다.
5. 개인용 액세스 토큰 페이지에서 새 토큰 생성을 선택합니다.
6. 새 개인용 액세스 토큰 페이지에서 노트에 토큰을 설명하는 이름을 입력합니다.
7. 범위 선택 섹션에서 admin:repo_hook을 선택합니다.
8. 토큰 생성을 선택합니다.
9. 개인용 액세스 토큰을 복사하고 저장합니다. CLI AWS CloudFormation또는 SDKs.

Amplify GitHub 앱이 GitHub 계정에 설치되고 개인 액세스 토큰을 생성한 후 Amplify CLI AWS CloudFormation또는 SDKs. `accessToken` 필드를 사용하여 이전 절차에서 생성한 개인용 액세스 토큰을 지정합니다. 자세한 내용은 Amplify API 참조의 CreateApp 및 AWS CloudFormation 사용 설명서의 AWS::Amplify::앱을 참조하십시오.

다음 CLI 명령은 리포지토리 액세스를 위해 GitHub 앱을 사용하는 새로운 Amplify 앱을 배포합니다. *myapp-using-Githubapp*, *https://Github.com/Myaccount/react-app*, *MY_TOKEN*을 사용자 고유의 정보로 바꾸십시오.

```
aws amplify create-app --name myapp-using-githubapp --repository https://github.com/
Myaccount/react-app --access-token MY_TOKEN
```

Amplify GitHub 앱을 사용하여 웹 미리 보기 설정하기

웹 미리보기는 GitHub 리포지토리에 대한 모든 풀 요청(PR)을 고유한 미리 보기 URL에 배포합니다. 프리뷰는 이제 Amplify GitHub 앱을 사용하여 GitHub 리포지토리에 액세스할 수 있습니다. 웹 미리 보

기용 GitHub 앱 설치 및 권한 부여에 대한 지침은 풀 요청에 대한 웹 미리 보기 활성화 단원을 참조하십시오.

AWS Amplify 호스팅 참조

이 섹션의 주제를 사용하여 자세한 참조 자료를 찾을 수 있습니다 AWS Amplify.

주제

AWS CloudFormation 지원

AWS CloudFormation 템플릿을 사용하여 Amplify 리소스를 프로비저닝하면 반복 가능하고 안정적인 웹 앱 배포가 가능합니다.는 클라우드 환경의 모든 인프라 리소스를 설명하고 프로비저닝할 수 있는 공통 언어를 AWS CloudFormation 제공하며 클릭 몇 번으로 여러 AWS 계정 및/또는 리전에서 롤아웃을 간소화합니다.

Amplify Hosting은 Amplify CloudFormation 설명서를 참조하십시오. Amplify Studio는 Amplify UI Builder CloudFormation 설명서를 참조하십시오.

AWS Command Line Interface 지원

AWS Command Line Interface 를 사용하여 명령줄에서 프로그래밍 방식으로 Amplify 앱을 생성합니다. 자세한 내용은 AWS CLI 설명서을(를) 참조하십시오.

리소스 태그 지정 지원

를 사용하여 Amplify 리소스 AWS Command Line Interface 에 태그를 지정할 수 있습니다. 자세한 내용은 AWS CLI 태그-리소스 설명서을(를) 참조하십시오.

Amplify Hosting API

이 참조는 Amplify Hosting API의 작업과 데이터 유형에 대한 설명을 제공합니다. 자세한 내용은 Amplify API 참조 문서를 참조하십시오.

에 대한 문서 기록 AWS Amplify

다음 표에서는의 마지막 릴리스 이후 설명서에 대한 중요한 변경 사항을 설명합니다 AWS Amplify.

- 최종 설명서 업데이트: 2025년 5월 28일

변경 사항	설명	날짜
빌드 설정 및 구성 장 업데이트	애플리케이션에 필요한 CPU, 메모리 및 디스크 공간 리소스를 제공하는 인스턴스 유형을 선택할 수 있는 새로운 구성 가능한 빌드 인스턴스 유형 기능을 설명하도록 Amplify 애플리케이션의 빌드 구성 관리장을 업데이트했습니다.	2025년 5월 28일
방화벽 장 업데이트	GA 기능 및 요금 구조를 AWS WAF포함하여 Amplify와 통합의 일반 가용성(GA)을 설명하도록 Amplify 호스팅 사이트에 대한 방화벽 지원장을 업데이트했습니다.	2025년 3월 26일
새로운 스큐 보호 장	Amplify 웹 애플리케이션의 클라이언트와 서버 간 버전 스큐 문제를 제거하는 스큐 보호 기능을 설명하는 Amplify 배포에 대한 스큐 보호장이 추가되었습니다.	2025년 5월 10일
업데이트된 Webhooks 장	단일 Git 리포지토리와 연결된 모든 Amplify 애플리케이션에 대해 하나의 포괄적인 웹후크를 사용하는 통합 웹후크 기능을 설명하는 Git 리포지토리용	2025년 5월 10일

변경 사항	설명	날짜
	통합 웹후크 주제를 추가했습니다.	
새로운 AWS 리소스에 대한 액세스를 허용하는 SSR 컴퓨팅 역할 추가 주제	Amplify SSR Compute 역할을 생성하고 앱과 연결하여 Amplify Compute 서비스에 리소스에 AWS 대한 액세스 권한을 부여하는 방법을 설명하는 AWS 리소스에 대한 액세스를 허용하는 SSR 컴퓨팅 역할 추가 주제가 추가되었습니다.	2025년 2월 17일
Amplify 앱 보호 장 AWS WAF 에 새로운 사용	웹 액세스 제어 목록 AWS WAF (웹 ACL)으로 웹 애플리케이션을 보호할 수 있는 Amplify와 (평가판에서)의 통합을 설명하는 Amplify 호스팅 사이트에 대한 방화벽 지원장이 추가되었습니다.	2024년 12월 18일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2024년 11월 14일
Next.js 주제에 대한 Amplify 지원 업데이트	Next.js 버전 15에 대한 Amplify의 지원을 설명하도록 Next.js 에 대한 Amplify 지원 주제를 업데이트했습니다.	2024년 11월 6일

변경 사항	설명	날짜
새로 추가된 Amazon S3 버킷에서 Amplify로 정적 웹 사이트 배포 장	몇 번의 클릭으로 S3에 저장된 정적 웹 사이트 콘텐츠를 호스팅할 수 있는 Amplify의 새로운 Amazon S3 통합을 설명하는 Amazon S3 버킷에서 Amplify에 정적 웹 사이트 배포 장을 추가했습니다.	2024년 10월 16일
새로 추가된 캐시 구성 관리 장	Amplify의 기본 캐싱 동작 및 관리형 캐시 정책을 콘텐츠에 적용하는 방법을 설명하는 앱의 캐시 구성 관리 장을 추가했습니다.	2024년 8월 13일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2024년 7월 18일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2024년 5월 31일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2024년 4월 17일

변경 사항	설명	날짜
업데이트된 시작하기 장	자습서에서 Next.js 예제 애플리케이션을 사용하도록 Amplify Hosting에 앱 배포 시작하기 장을 업데이트했습니다.	2024년 4월 12일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2024년 4월 5일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2024년 4월 4일
새로 추가된 문제 해결 장	Amplify Hosting에 배포된 애플리케이션에서 발생하는 문제를 해결하는 방법을 설명하는 Amplify Hosting 문제 해결 장을 추가했습니다.	2024년 4월 2일
사용자 지정 SSL/TLS 인증서에 대한 새로운 지원	앱을 사용자 지정 도메인에 연결할 때 Amplify의 사용자 지정 SSL/TLS 인증서에 대한 지원을 설명하는 SSL/TLS 인증서 사용 항목을 사용자 지정 도메인 연결 장에 추가했습니다.	2024년 2월 20일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2024년 1월 2일

변경 사항	설명	날짜
SSR 프레임워크를 위한 새로운 지원	오픈 소스 어댑터를 사용하는 JavaScript 기반 SSR 프레임워크를 위한 Amplify 지원을 설명하는 Amplify Hosting을 통해 서버 측 렌더링 애플리케이션 배포 항목을 업데이트했습니다.	2023년 11월 19일
이미지 최적화 기능 시작을 위한 새로운 지원	서버 측에서 렌더링한 앱의 이미지 최적화에 기본적으로 제공되는 지원을 설명하는 SSR 앱을 위한 이미지 최적화 주제를 추가했습니다.	2023년 11월 19일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2023년 11월 17일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2023년 11월 6일
새 와일드카드 하위 도메인 항목	사용자 지정 도메인의 와일드카드 하위 도메인 지원을 설명하는 와일드카드 하위 도메인 설정 항목이 추가되었습니다.	2023년 11월 6일

변경 사항	설명	날짜
새로운 관리형 정책	Amplify에 대한 새로운 AmplifyBackendDeployFullAccess AWS 관리형 정책을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 주제를 업데이트했습니다.	2023년 10월 8일
모노레포 프레임워크 기능 출시에 대한 새로운 지원	npm 작업 공간, pnpm 작업 공간, Yarn 작업 공간, Nx 및 Turborepo를 사용하여 만든 모노리포지토리에 앱을 배포하기 위한 지원을 설명하도록 모노리포지토리 빌드 설정 구성 항목을 업데이트했습니다.	2023년 6월 19일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2023년 6월 1일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2023년 2월 24일
서버 측 렌더링 챕터 업데이트	Next.js 버전 12 및 13에 대한 Amplify 지원의 최근 변경 사항을 설명하도록 Amplify Hosting을 통해 서버 측 렌더링 애플리케이션 배포 장을 업데이트했습니다.	2022년 11월 17일

변경 사항	설명	날짜
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2022년 8월 30일
업데이트된 관리형 정책 항목	Amplify Studio를 사용하여 백엔드를 배포하는 방법을 설명하도록 애플리케이션의 백엔드 빌드 항목을 업데이트했습니다.	2022년 8월 23일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2022년 4월 27일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2022년 4월 17일
새로운 GitHub 앱 기능 출시	GitHub 리포지토리에 대한 Amplify 액세스를 승인하기 위한 새로운 GitHub 앱을 설명하는 GitHub 리포지토리에 대한 Amplify 액세스 설정 항목이 추가되었습니다.	2022년 4월 5일

변경 사항	설명	날짜
Amplify Studio의 새로운 기능 출시	백엔드 데이터에 연결할 수 있는 UI 구성 요소를 만들 수 있는 비주얼 디자이너를 제공하는 Amplify Studio 업데이트를 설명하도록 AWS Amplify 호스팅 시작 항목을 업데이트했습니다.	2021년 12월 2일
업데이트된 관리형 정책 항목	Amplify Studio를 지원하기 위한 Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2021년 12월 2일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2021년 11월 8일
업데이트된 관리형 정책 항목	Amplify의 AWS 관리형 정책에 대한 최근 변경 사항을 설명하도록 AWS 에 대한 관리형 정책 AWS Amplify 항목을 업데이트했습니다.	2021년 9월 27일
새로운 관리형 정책 항목	Amplify에 대한 AWS 관리형 정책과 해당 정책에 대한 최근 변경 사항을 설명하는 AWS 에 대한 관리형 정책 AWS Amplify 주제를 추가했습니다.	2021년 7월 28일

변경 사항	설명	날짜
서버측 렌더링 챕터가 업데이트되었습니다.	Next.js 버전 10.x. x 및 Next.js 버전 11에 대한 새로운 지원을 설명하도록 Amplify Hosting을 통해 서버 측 렌더링 애플리케이션 배포 장을 업데이트했습니다.	2021년 7월 22일
빌드 설정 구성 챕터 업데이트	Amplify를 사용하여 monorepo 앱을 배포할 때 빌드 설정 및 새로운 `AMPLIFY_MONOREPO_APP_ROOT` 환경 변수를 구성하는 방법을 설명하는 모노 리포지토리 빌드 설정 구성 항목이 추가되었습니다.	2021년 7월 20일
기능 브랜치 배포 챕터가 업데이트되었습니다.	빌드 시 `aws-exports.js` 파일을 자동 생성하는 방법을 설명하는 Amplify 구성의 자동 빌드 타임 생성(Gen 1 앱만 해당) 항목이 추가되었습니다. 조건부 백엔드 빌드를 활성화하는 방법을 설명하는 조건부 백엔드 빌드(Gen 1 앱만 해당) 항목이 추가되었습니다. 새 앱을 만들거나, 새 브랜치를 기존 앱에 연결하거나, 다른 백엔드 환경을 가리키도록 기존 프론트엔드를 업데이트할 때 기존 백엔드를 재사용하는 방법을 설명하는 앱 전체에서 Amplify 백엔드 사용(Gen 1 앱만 해당) 항목이 추가되었습니다.	2021년 6월 30일

변경 사항	설명	날짜
업데이트된 보안 장	공동 책임 모델을 적용하는 방법과 Amplify가 암호화를 사용하여 저장된 데이터와 전송 중인 데이터를 보호하는 방법을 설명하는 Amplify의 데이터 보호 항목을 추가했습니다.	2021년 6월 3일
SSR 기능 출시에 대한 새로운 지원	서버 사이드 렌더링(SSR)을 사용하고 Next.js로 만든 웹 앱에 대한 Amplify 지원을 설명하는 Amplify Hosting을 통해 서버 측 렌더링 애플리케이션 배포 장을 추가했습니다.	2021년 5월 18일
새로운 보안 장	Amplify를 사용할 때 공동 책임 모델을 적용하는 방법과 보안 및 규정 준수 목표를 충족하도록 Amplify를 구성하는 방법을 설명하는 Amplify의 보안 장을 추가했습니다.	2021년 3월 26일
업데이트된 사용자 지정 빌드 항목	Amazon Elastic Container Registry Public에서 호스팅되는 사용자 지정 빌드 이미지를 구성하는 방법을 설명하기 위해 사용자 지정 빌드 이미지 및 라이브 패키지 업데이트 항목을 업데이트했습니다.	2021년 3월 12일
모니터링 항목 업데이트	Amazon CloudWatch 지표 데이터에 액세스하고 경보를 설정하는 방법을 설명하도록 모니터링 항목을 업데이트했습니다.	2021년 2월 2일

변경 사항	설명	날짜
CloudTrail 로깅 항목 새로 추가	가 AWS Amplify 콘솔 API 참조 및 관리자 UI API 참조에 대한 모든 API 작업을 캡처하고 기록하는 방법을 설명하는 주제를 사용하여 Amplify API 직접 호출 로깅 AWS CloudTrail을 추가했습니다. AWS CloudTrail AWS Amplify	2021년 2월 2일
새 관리자 UI 기능 출시	프런트엔드 웹 및 모바일 개발자가 AWS Management Console외부에서 앱 백엔드를 만들고 관리할 수 있는 시각적 인터페이스를 제공하는 새로운 관리 UI를 설명하도록 AWS Amplify 호스팅 시작 항목을 업데이트했습니다.	2020년 12월 1일
새 성능 모드 기능 출시	성능 모드를 활성화하여 더 빠른 호스팅 성능을 위해 최적화하는 방법을 설명하도록 앱 성능 관리 항목을 업데이트했습니다.	2020년 11월 4일
사용자 지정 헤더 항목을 업데이트했습니다.	콘솔을 사용하거나 YML 파일을 편집하여 Amplify 앱의 사용자 지정 헤더를 정의하는 방법을 설명하도록 사용자 지정 헤더 항목을 업데이트했습니다.	2020년 10월 28일

변경 사항	설명	날짜
새로운 자동 하위 도메인 기능 출시	Amazon Route 53 사용자 지정 도메인에 연결된 앱에 패턴 기반 기능 분기 배포를 사용하는 방법을 설명하는 Route 53 사용자 지정 도메인의 자동 하위 도메인 설정 항목을 추가했습니다. 하위 도메인에서 액세스할 수 있도록 pull 요청의 웹 미리 보기를 설정하는 방법을 설명하는 하위 도메인을 통한 웹 미리 보기 액세스 항목을 추가했습니다.	2020년 6월 20일
새 알림 항목	빌드의 성공 또는 실패 시 이해관계자 또는 팀 구성원에게 알리도록 Amplify 앱에 이메일 알림을 설정하는 방법을 설명하는 알림 항목이 추가되었습니다.	2020년 6월 20일
사용자 지정 도메인 항목을 업데이트했습니다.	Amazon Route 53, GoDaddy 및 Google 도메인에 사용자 지정 도메인을 추가하는 절차를 개선하도록 사용자 지정 도메인 연결 항목을 업데이트했습니다. 이 업데이트에는 사용자 지정 도메인 설정을 위한 새로운 문제 해결 정보도 포함되어 있습니다.	2020년 5월 12일
AWS Amplify 릴리스	이번 릴리스에서 Amplify가 도입되었습니다.	2018년 11월 26일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.